Hochschule Offenburg
offenburg.university

Bachelor Thesis

Untersuchung und Vergleich von Open Source Plattformen für das Smart Home

Fakultät
Medien und Informationswesen

Studiengang
Medien und Informationswesen

vorgelegt von

Pirmin Gersbacher

Wintersemester 2017/2018

Erstprüfer: Prof. Dr. Rüdebusch
Zweitprüfer: Prof. Dr. Sänger

Eigenständigkeitserklärung

Hiermit bestätigen ich, dass die vorliegende Abschulussarbeit von mir selbstständig verfasst wurde und keine anderen als die angegebenen Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Datum: _____ Unterschrift: _____

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

IoT Internet of Things

PaaS Platform as a Service

BAALL Bremen Ambient Assisted Living Lab

MIT Massachusetts Institute of Technology

OSI Open Source Initiative

OSD Open Source Definition

IFTTT If This Then That

HTTP Hypertext Transfer Protocol

BLE Bluetooth Low Energy

MQTT Message Queue Telemetry Transport

SoC System on a Chip

.

1 Vorwort

Das Licht wird gedimmt, sobald ein Film gestartet wird, die Jalousien schließen sich automatisch, wenn die Mittagssonne die Zimmer des Hauses aufheizt. Die Vernetzung verschiedener Geräte im Haus erhöht den Komfort, spart Energie und in vielen Haushalten befinden sich bereits intelligente Elektronikgeräte. Doch meist arbeiten diese als Insellösungen unabhängig aneinander vorbei. Funkthermostat, smarte Glühbirne und Funksteckdose benötigen zur Steuerung jeweils ihre eigenen Apps oder Fernbedienungen, programmierte Abläufe und Interaktionen zwischen diesen intelligenten Helfern finden nicht statt.

Smart Home Lösungen versprechen hier Abhilfe. Hier gibt es zum einen sogenannte proprietäre, also geschlossene, herstellerabhängige Systeme. Diese lassen allerdings meist nur wenige Erweiterungen für den individuellen Zweck zu und man ist zumeist an die smarten Geräte von diesem speziellen Hersteller gebunden. Demgegenüber stehen Open-Source-Softwarelösungen, wie beispielsweise OpenHAB, ioBroker, Home Assistant und Node-RED. Diese bieten die Möglichkeit, unterschiedliche Geräte von unterschiedlichsten Herstellern in einer Anwendung zu vereinen und so verschiedene Heim-Automatisierungssysteme unter einen Hut zu bringen.

Das reizvolle an diesen Anbietern ist zusätzlich: sie sind kostenlos, quelloffen und eine große Community von Nutzern erweitert das Spektrum an eingebundenen Geräten und neuen Funktionalitäten kontinuierlich und stellt diese anderen Anwendern zur Verfügung.

Im Rahmen dieser Bachelor-Thesis findet eine Auseinandersetzung mit den vier Open-Source-Plattformen zur zentralen Verwaltung von intelligenten Geräten im Smart Home OpenHAB, ioBroker, Node-RED und Home Assistant statt, um diese, anhand zuvor festgelegter Kriterien, in einem repräsentativen Smart Home Szenario zu vergleichen.

Dafür wird zunächst anhand einer Literaturrecherche genauer auf die Begriffe Smart Home, Open Source und die jeweiligen Open-Source-Plattformen eingegangen. Anschließend werden Vergleichskriterien für die verschiedenen Plattformen festgelegt und Anforderungen an ein exemplarisches und repräsentatives Smart Home Projekt definiert.

Im letzten Schritt wird das zuvor definierte Smart Home Szenario mit jeder der zu vergleichenden Plattformen realisiert, um so die unterschiedlichen Plattformen miteinander zu vergleichen und zu bewerten.

2 Das Internet der Dinge und Smart Home

2.1 Das Internet der Dinge

Es ist ein riesiges Netz, das sich bereits aufgebaut hat und weiterhin rasant wächst - das Internet der Dinge (engl. Internet of Things (IoT)). Zu der Menge der bereits vorhandenen IoT-Devices gibt es unterschiedliche Zahlen. Dies liegt unter anderem daran, dass die genutzten IoT-Geräte nirgendwo angemeldet werden und so ihre Anzahl nur geschätzt werden kann. So beziffert die International Data Corporation, ein international tätiges Marktforschungsunternehmen, die Anzahl der im Jahr 2016 vorhandenen IoT-Devices auf etwa 9 Mrd. Hier sind Smartphones, Tablets und Computer noch nicht miteinberechnet [19]. Gartner Inc., ein weiterer großer Anbieter von Marktforschungsergebnissen im Bereich IT, kommt auf einen ähnlichen, allerdings etwas geringeren Wert und schätzt, dass die Anzahl der miteinander vernetzten Dinge im Internet noch im Jahr 2017 auf 8,4 Mrd. steigen wird [20].

Das Konzept eines Internets der Dinge existiert bereits seit Jahrzehnten. So wurde beispielsweise schon im Jahr 1982 ein Cola-Automat von Studenten der Carnegie Mellon University mit Sensoren ausgestattet und mit dem Internet verbunden. Über den Befehl *finger cokecs.cmu.edu* konnte jeder im Internet herausfinden, wie viele Getränkedosen sich noch in dem Automaten befinden. Noch bis Januar 2014 ist dokumentiert, dass sich der Cola-Automat weiterhin im 6. Stockwerk des Gates-Gebäudes der Carnegie Mellon University befand [21] [22].

Kevin Ashton, einem Forscher des Massachusetts Institute of Technology (MIT), wird nachgesagt, im Jahr 1999 als erstes den Begriff Internet of Things verwendet zu haben. Er selbst erklärt das IoT so, dass früher Computer und Internet für die Gewinnung von Informationen fast vollständig vom Menschen abhängig waren. Im 20. Jahrhundert waren Computer so „brains whithout senses“ – also Gehirne ohne Sinnesorgane. [23].

Die Daten im Internet wurden von Menschen erfasst und erstellt, beispiels-

Abbildung 2.1: Coke Machine von Studenten der CMU [1]

weise mittels Eintippen über die Tastatur oder Drücken eines Knopfes. Durch das Internet der Dinge wird es Computern ermöglicht, selbst Informationen zu sammeln. Man ermöglicht ihnen die Welt zu „sehen“, zu „hören“ und zu „riechen“. Mittels Sensoren können Computer die Welt beobachten, sie erkennen und sie verstehen, unabhängig vom Menschen und dessen begrenzter Zeit für das Sammeln und Eingeben von Daten [24].

Der wissenschaftliche Dienst des deutschen Bundestages holt bei seiner Festlegung des Begriffs Internet der Dinge noch etwas weiter aus und beruft sich dabei auf eine, durch die deutsche EU-Ratspräsidentschaft im Jahr 2007 entwickelte, Definition. So wird das Internet der Dinge als eine technische Vision beschrieben, in welcher Objekte jeder Art in einem digitalen Netz integriert werden. Mittels einer eindeutigen Identität können Alltagsgegenstände über das Netz angesteuert werden und diese so selbstständig miteinander kommunizieren. So wird eine Verbindung zwischen der physischen Welt der Dinge und der virtuellen Welt der Daten geschaffen. Durch Mikroprozessoren und Sensoren sind Dinge so in der Lage, ihre Umgebung wahrzunehmen, Informationen zu gewinnen und zu verarbeiten und mit anderen Objekten zu kommunizieren

und dabei auch selbst Aktionen auszulösen. Alltägliche Gegenstände werden zu intelligenten Objekten [25].

Cisco beschreibt in einem Paper aus dem Jahr 2011 das Internet der Dinge als die nächste Evolution des Internets. Während sich das Internet zuvor zwar stetig weiterentwickelt hat, hat es für eine lange Zeit grundsätzlich noch das selbe gemacht, wie zu seiner Anfangszeit als „Advanced Research Projects Agency Network“ (ARPANET). So gab es bereits zu jener Zeit verschiedene, noch heute eingesetzte Protokolle, wie beispielsweise IP. Das Internet der Dinge hat das Potential die Art, wie die Menschen leben, lernen und arbeiten, dramatisch zu verbessern [26].

Eine der interessantesten Entwicklungen, die vom Internet der Dinge abstammt, ist die Heim-Automatisierung bzw. das Smart Home, worauf im Folgenden nun genauer eingegangen wird.

2.2 Smart Home

2.2.1 Definition

Das Smart Home stellt für den normalen Anwender den vielleicht „relevantesten und unmittelbar erfahrbarsten Anwendungsbereich des ‚Internets der Dinge‘ dar“ [27]. Das Internet der Dinge wird durch das intelligente Zuhause in den Alltag integriert und tagtäglich erlebt.

Strese et al. definieren das Smart Home als ein privat genutztes Heim, in dem verschiedene Geräte aus der Hausautomation, wie etwa Heizung oder Beleuchtung, aus der Haushaltstechnik (z.B. Kühlschrank oder Waschmaschine) oder aus der Konsumelektronik, zu intelligenten Gegenständen werden. Diese Gegenstände orientieren sich an den Bedürfnissen der Bewohner. Untereinander vernetzt können diese so dem Nutzer Assistenzfunktionen und Dienste bereitstellen [28].

Eine weitere Deutung definiert das Smart Home als eine Wohnstätte, ausgestattet mit Rechnern und technologischen Geräten, welche die Bedürfnisse der Bewohner antizipieren und auf diese reagieren. Durch die Steuerung der Technologie innerhalb des Hauses und der Verbindung zur Außenwelt werden Komfort, Unterhaltung und Sicherheit gesteigert [29].

Ein Smart Home besteht also aus vernetzten Geräten innerhalb des Wohnbereichs, zwischen denen automatisierte Abläufe stattfinden. Die Geräte interagieren mit dem Bewohner, werten Daten aus und treffen intelligente Entschei-

dungen und unterstützen den Nutzer auf diese Weise.

2.2.2 Historische Entwicklung

So genannte „wiring homes“ gab es bereits in den 1960er-Jahren. Diese verkabelten und so mit mehr Funktionalität ausgestatteten Häuser, können durchaus bereits als „smart“ bezeichnet werden und stellen eine Art Vorgänger des Smart Homes dar. Allerdings beschäftigten sich damals nur Hobby-Bastler in diesem Bereich. Das Wort Smart Home wurde erstmals offiziell im Jahr 1984 genutzt und zwar von der „American Association of House Builders“[29].

Seit den 80er-Jahren entwickeln Hersteller für Unterhaltungselektronik Systeme und Komponenten für den Hausgebrauch [29]. Im Jahr 1985 stellte beispielsweise das US-Unternehmen Unity Systems erstmals den so genannten „Home Manager“ her. Mit diesem konnte unter anderem die Temperatur in den Räumen des Hauses an Uhrzeit und Wochentag angepasst werden. Ebenso ließen sich Beleuchtung, Alarmanlage und HiFi-Anlagen kontrollieren. Steuerbar war dies alles über einen Röhrenbildschirm mit Touchscreen-Interface [2].

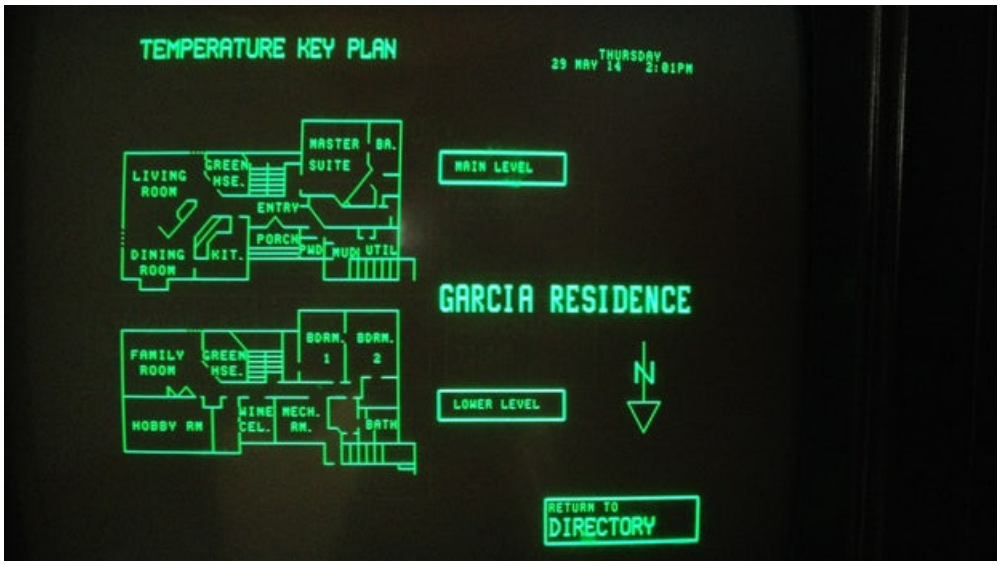


Abbildung 2.2: 80er Jahre Smart-Home-System [2]

Im Verlauf der 1990er-Jahre erlangte das Konzept des Smart Homes immer mehr Akzeptanz in der breiten Bevölkerung. Artikel über das Smart Home erschienen in unterschiedlichen Life-Style-Magazinen, wie beispielsweise der Vanity Fair. Allerdings wurde das intelligente Haus auch weiterhin mit einer gewissen Skepsis betrachtet, speziell aufgrund des Kontrollverlustes, der mit der

Automatisierung des Hauses einhergeht [29].

Heute ist das Konzept des Smart Homes allgegenwärtig und weit akzeptiert. Bei der Internationalen Funkausstellung (IFA), der weltweit führenden Messe für Consumer Electronics, gab es 2016 erstmals einen Smart Home Ausstellungsbereich [30]. Große Unternehmen, wie beispielsweise Google, Telekom und RWE, kämpfen um eine gute Position im Smart Home Markt. So übernahm Google im Jahr 2014 für 3,2 Mrd. Dollar den Hersteller für vernetzte Haustechnik Nest Labs und unterstrich so seine Ambitionen im Smart Home Markt [31]. Eine repräsentative Umfrage mit über 1.000 Teilnehmern des Marktforschungsunternehmens Splendid Research kam zu dem Ergebnis, dass mehr als drei von vier Deutschen zur potentiellen Nutzergruppe für Smart Home zählen oder bereits Nutzer von Smart Home Geräten sind [32].

2.2.3 Smart Home Architektur

Ein Smart Home besteht aus unterschiedlichen Komponenten. Diese sind Sensoren, Controller, Aktoren, das Steuernetz sowie Gateways.

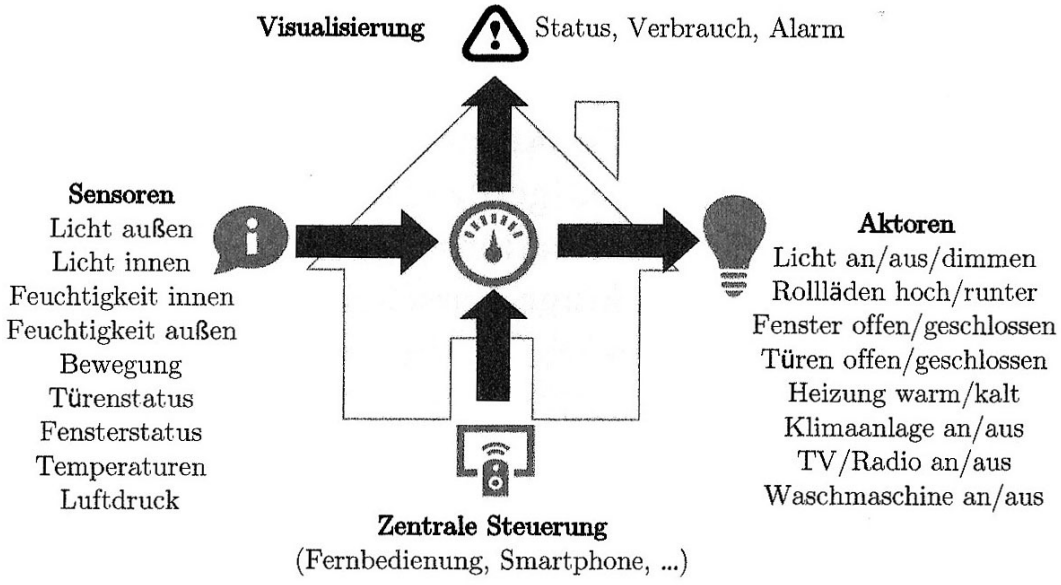


Abbildung 2.3: Smart Home Architektur [3]

Sensoren

Sensoren nehmen Signale wahr und generieren auf diese Weise Informationen. Ein Beispiel für einen Sensor ist der Temperatursensor im Raumthermostat. Die gesammelten Informationen werden an andere Geräte mittels des Kommunikationsnetzwerkes übertragen [3].

Steuernetz

Bei dem Steuernetz handelt es sich um ein Kommunikationsmedium, das Sensoren, Aktoren und Controller miteinander verbindet. Meist handelt es sich hierbei um ein drahtloses Funksystem, es kann sich allerdings auch um eine Verkabelung handeln [3].

Aktoren

Aktoren führen Aktionen aus. Bei diesen Aktionen kann es sich etwa um das Aus- oder Einschalten eines Gerätes handeln. Typische Aktoren sind Lichtschalter oder Fenstermotoren [3].

Controller

Controller steuern die Aktoren des Smart Homes mittels des Kommunikationsnetzwerkes. Die Intelligenz des Steuernetzes des smarten Zuhauses befindet sich im Zentralcontroller. Der Controller verarbeitet die Informationen der Sensoren und Aktoren und führt automatisierte Prozesse aus [3]. Bei Controllern kann es sich allerdings ebenso um Bedienoberflächen handeln, an welchen der Bewohner Informationen erhält und Aktoren ansprechen kann.

Gateways

Über Gateways kann das Kommunikationsnetzwerk des Hauses mit anderen Kommunikationsnetzwerken, wie beispielsweise dem Internet, verbunden werden [3].
Hierbei kann es sich etwa um die Anbindung von Cloud-Diensten wie If This Then That (IFTTT) handeln (vgl. Abbildung 2.7).

2.2.4 Anwendungsgebiete

Das Smart Home, selbst Teilbereich des Internets der Dinge, umfasst eine Vielzahl von Teilsystemen der Hausautomatisierung. Beispiele dafür sind die Heizungssteuerung, Lüftung bzw. Klima, das Energiemanagement, die Lichtsteuerung durch Lichtmanagement und Steuerung der Rollos, die Überwachung technischer Art z.B. in Form von Rauchmeldern oder auch der Schutz gegen Einbruch und die Zutrittskontrolle [28].

Zusätzlich umfasst das Smart Home auch direkt auf die Bewohner bezogene Produkte wie die Konsumelektronik (TV, Audio, etc.), Hausgeräte wie Kühlschrank und Waschmaschine oder auch Geräte zur medizinischen Diagnostik und Vorsorge oder zur Betreuung von pflegebedürftigen Menschen. Das Hauptziel dabei ist, dass die Teilsysteme die Aufgaben, die ihnen gestellt werden, möglichst autonom erledigen und sich dafür auch mit anderen Komponenten austauschen [28].

Folgende Grafik gibt eine Übersicht zu möglichen Anwendungsfeldern:

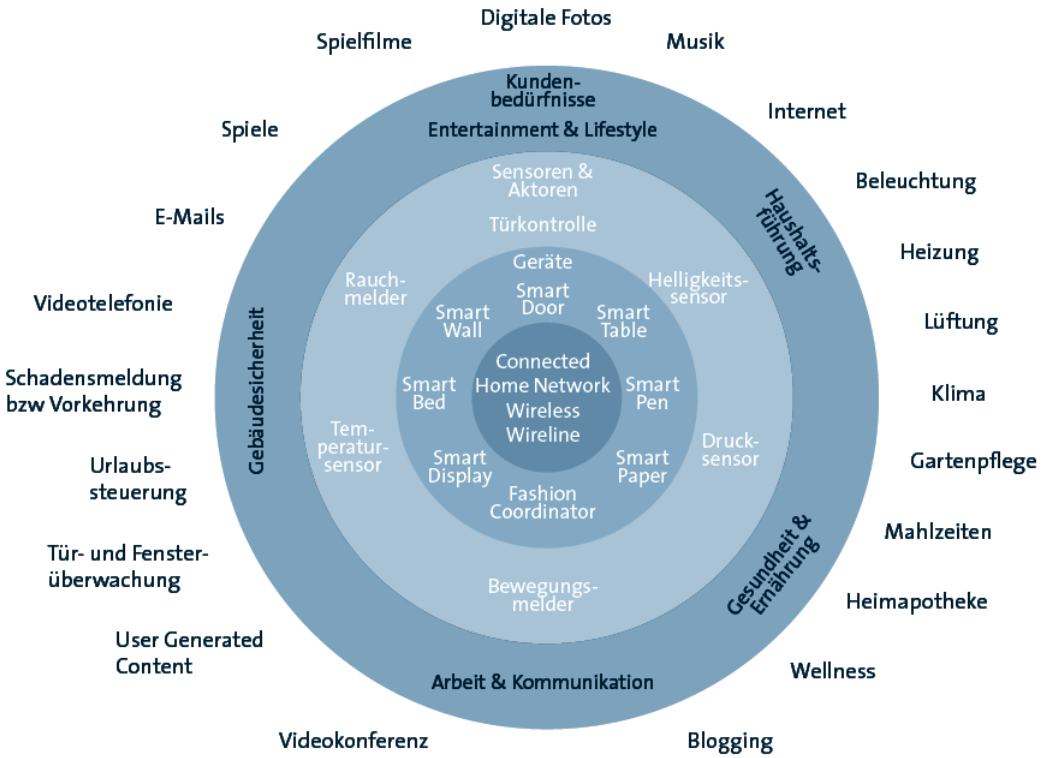


Abbildung 2.4: Anwendungsfelder im Smart Home [4]

2.2.5 Kommunikationsstandards

Kommunikationsstandards legen fest, wie die einzelnen Geräte im Smart Home untereinander und mit dem Anwender kommunizieren. Die Kommunikation im Smart Home erfolgt entweder kabelgebunden oder per Funk. Im Smart Home Markt kämpfen verschiedene Technologien um Marktanteile. Speziell drahtlose Funktechnologien ermöglichen es, ein Smart Home flexibel, komfortabel und kostengünstig einzurichten [33].

Protokollschichten

Aufgrund der komplexen Anforderungen von Computernetzen wird die Kommunikation in diesen auf unterschiedliche Schichten unterteilt. Jede der Schichten geht auf einen bestimmten Gesichtspunkt der Kommunikation ein und stellt Schnittstellen zur darüber- und darunterliegenden Schicht bereit. Jede Schicht fügt zu einer Nachricht weitere Zusatzinformationen hinzu. Diese Informationen werden Header genannt [5]. In der Literatur wird diese Kommunikation häufig im Hybriden Referenzmodell als fünfschichtiges Modell dargestellt.

Abbildung 2.5: Hybrides Referenzmodell [5]

Die unterste Schicht ist die Bitübertragungsschicht. In dieser findet der physische Anschluss an das Medium und die Umwandlung der Daten in Signale statt. Hier wird also die Übertragung der Einsen und Nullen umgesetzt [5].

Über der Bitübertragungsschicht befindet sich die Sicherungsschicht. Sie ist für die Fehlererkennung bei der Übertragung zuständig. Hierzu werden beim Sender in der Sicherungsschicht die Datenpakete in Frames verpackt. Anschließend

werden diese, mit der ausgewählten Zuverlässigkeit, innerhalb des physischen Netzes von einem Gerät zum anderen übertragen. Zur Übertragung werden physische Adressen (MAC-Adressen) benötigt, welche von den Protokollen dieser Schicht jedem der Frames, zusammen mit einer Prüfsumme, hinzugefügt werden. Anhand der Prüfsumme können fehlerhafte Frames erkannt und verworfen werden [5].

Die Protokolle der Vermittlungsschicht sind dafür zuständig, dass die Daten bei der Kommunikation in einem logischen Netz über physische Übertragungsabschnitte hinweg weitervermittelt werden. Hierzu werden logische Adressen (IP-Adressen) verwendet. Zumeist wird das Internetprotokoll IP genutzt, bei dem jedes der Datenpakete unabhängig an das Ziel vermittelt wird [5].

Mittels der Transportschicht wird der Transport zwischen Prozessen auf unterschiedlichen Geräten, über Ende-zu-Ende-Protokolle, realisiert. Beim Sender verpacken die Protokolle der Transportschicht die Daten der Anwendungsschicht in so genannte Segmente. Für die Adressierung werden Portnummern verwendet. So wird dafür gesorgt, dass die Daten von der Vermittlungsschicht an die gewünschte Anwendung ausgeliefert werden. Typische Transportschicht-Protokolle sind TCP und UDP [5].

Die Protokolle der Anwendungsschicht arbeiten mit den Anwendungsprogrammen, wie beispielsweise dem Browser, zusammen. In diesen befindet sich die eigentliche Nachricht, also beispielsweise eine HTML-Seite. Typische Protokolle der Anwendungsschicht sind HTTP, MQTT, FTP und SMTP [5].

Einige der gängigsten Kommunikationsstandards sollen im Folgenden kurz vorgestellt werden.

WLAN/Wi-Fi

Wi-Fi ist die Funktechnologie, die am Markt am meisten vertreten ist. In nahezu jedem Zuhause ist WLAN zu finden. Diese Technologie ist allerdings nicht für die Hausautomation optimiert und wurde deshalb auch nicht zum Smart Home Standard.

WLAN wurde für den Transport großer Datenmengen entwickelt. Der Fokus liegt hier auf der Geschwindigkeit und der Übermittlung der großen Datenmengen. Dies hat Auswirkungen auf den Energieverbrauch. Für die Hausautomation, bei welcher meist auf Geräte zurückgegriffen wird, die batteriebetrieben sind, ist WLAN als Standard deshalb eher ungeeignet. Es kann nur bei Komponenten im automatisierten Haus genutzt werden, die ständig mit dem Stromnetz verbunden sind und so nicht alle Einsatzbereiche abdecken. WLAN

wird im Smart Home genutzt, um Geräte wie Smartphones oder Tablets bis zu einem Gateway einzubinden. Ab diesem wird das Signal allerdings mittels Kommunikationsstandards, die einen niedrigeren Energieverbrauch haben, an die jeweiligen Sensoren und Aktoren weitergeleitet [3].

ZigBee

Der Kommunikationsstandard ZigBee nutzt IEEE 802.15.4 als Funkschicht. Bei IEEE 802.15.4 handelt es sich um eine Kommunikationsverbindung, welche niedrigen Energieverbrauch und damit einhergehend eine niedrige Datenrate als Merkmal aufweist. Dieser Standard deckt allerdings nur die unteren beiden Protokollebenen des Hybriden Referenzmodells ab. Da es keine höheren Kommunikationsschichten gibt, kann diese Technik nicht als komplette Lösung für das Kommunikationsnetz dienen, wird allerdings häufig als zugrundeliegende Schicht für Kommunikationslösungen im Smart Home genutzt [3].

ZigBee ist eine Entwicklung der ZigBee Alliance, welche im Jahr 2002 gegründet wurde. Die erste ZigBee-Spezifikation kam im Jahr 2004 auf den Markt. Inzwischen umfasst die Allianz mehr als 400 Unternehmen aus 37 Ländern [3].

Aufgrund seines niedrigen Energieverbrauchs eignet sich dieser Standard ideal für die Nutzung von batteriebetriebenen Geräten [34]. Die Datenrate von bis zu 250 Kbit/s ist dabei bewusst niedrig und es kann über Entfernungen von bis zu 100 Metern kommuniziert werden. Zusätzlich können ZigBee-Netzwerke als Mesh-Topologie angeordnet sein [34]. Die einzelnen Komponenten im ZigBee-System sind so alle untereinander vernetzt und stehen in direkter Kommunikation. So können in ZigBee-Netzwerken Daten über große Distanzen übermittelt werden, indem ein Gerät die Information zum nächsten Gerät weiterleitet. Besonders bekannt ist der Standard auch daher, dass Philips für seine Hue-Lights ZigBee als Kommunikationstechnologie nutzt.

Z-Wave

Z-Wave ist einer der am weitverbreitetsten Kommunikationsstandards im Smart Home. Er zeichnet sich durch eine hohe Netzwerkverfügbarkeit und -stabilität aus [34].

Entwickelt wurde Z-Wave von dem Unternehmen Zensys im Jahr 2001. Die Z-Wave Alliance wurde im Jahr 2005 gegründet. 2008 wurde Zensys von Sigma Designs übernommen. Stand Januar 2017 befinden sich 550 Hersteller in dieser Vereinigung [3]. Alle Hersteller von Z-Wave Produkten müssen ihre Geräte

gewissen Tests unterziehen, um die Erlaubnis zu erhalten, diese als Z-Wave-Produkte bewerben zu dürfen.

Mit Z-Wave werden Übertragungsgeschwindigkeiten von etwa 40 Kbit/s und Übertragungsreichweiten von 20-30 Metern erreicht. Z-Wave Geräte können, ebenso wie ZigBee-Devices, in einer Mesh-Formation angeordnet werden. Auf diese Weise kann die Reichweite des Netzes stark erweitert werden [35].

Thread

Thread nutzt, ebenso wie ZigBee, IEEE 802.15.4 als Funkschicht. Die Thread Group wurde im Jahr 2014 um das Unternehmen Nest gegründet, welches 2012 von Google übernommen wurde. Erst im Jahr 2016 wurde die erste anwendbare Spezifikation des Protokolls publiziert. Thread ermöglicht anderen Herstellern, mit einer eigenen Anwenderschicht zu arbeiten, aufbauend auf der Netzwerkschicht von Thread. Die Netzwerkschicht von Thread beruht auf einem Mapping aus IPv6 Adressen auf IEEE 802.15.4 Datenpaketen [3].

EnOcean

Als Ableger der Siemens AG wurde die EnOcean GmbH im Jahr 2001 gegründet [3]. EnOcean produziert batterielose Schalt-, Sensor- und Empfangsmodule. Der batterielose Betrieb wird durch Energy-Harvesting ermöglicht. Miniaturisierte Energiewandler wandeln Bewegungen, Licht oder Temperaturdifferenzen in elektrische Energie um und versorgen so die Module. Für die Energiegewinnung wird beispielsweise der piezoelektrische Effekt genutzt. Das Drücken des Schalters reicht aus, um die notwendige Energie für die Übertragung eines Funksignals zu erzeugen. EnOcean-Geräte erreichen eine Funkreichweite von bis zu 30 Metern. Über 1500 Produkte existieren bereits im EnOcean-Ökosystem [36].

Die Kommunikation ist nicht so zuverlässig wie beispielsweise bei Z-Wave oder ZigBee, da die Energie, welche durch die alternative Energiegewinnung erzeugt wird, zu gering ist, um eine Zwei-Wege-Kommunikation zu ermöglichen [3]. Dafür ist EnOcean durch den batterielosen Betrieb weitestgehend wartungsfrei und die Module können auch an schwer zugänglichen Stellen angebracht werden. So wurden, nach eigenen Angaben, weltweit bereits mehrere hunderttausend Gebäude mit EnOcean-Funkmodulen ausgestattet [36].

Bluetooth

Bei Bluetooth handelt es sich um ein Funksystem für die Übertragung von Daten über kurze Distanzen. Ursprünglich wurde es entwickelt, um kurze Kabelverbindungen zwischen verschiedenen Geräten zu ersetzen [5].

Entwickelt wurde es zunächst von der schwedischen Firma Ericsson im Jahr 1994. Später nahm sich die Interessengemeinschaft Bluetooth Special Interest Group (BSIG) der Weiterentwicklung des Übertragungsstandards an [5]. Die Nutzung von Bluetooth-Geräten ist zulassungsfrei. Von Bluetooth sind bereits mehrere Protokollversionen erschienen. Die aktuellste Version ist Bluetooth 5.0. Jede neue Version ist zur vorherigen abwärtskompatibel [37].

Bluetooth Low Energy (BLE) ist eine energiesparende Version von Bluetooth mit relativ kurzer Reichweite, die mit Version 4.0 eingeführt wurde. Diese soll sich besonders für IoT-Lösungen eignen. Sensoren, welche mit BLE kommunizieren, sollen so mit einer einfachen Knopfbatterie bis zu zwei Jahre betrieben werden können [38].

Mit der neuesten Version Bluetooth 5.0 sollen hohe Übertragungsreichweiten und energiesparender Betrieb vereint werden. Mit Bluetooth 5.0 können, bereits in der Low Energy Version, maximale Datenraten von 2 Mbit/s und Reichweiten von 200 Metern erreicht werden. Zusätzlich bietet die neueste Version, ebenso wie ZigBee oder Z-Wave, eine Mesh-Spezifikation zur weitreichenden Vernetzung. Bluetooth-Geräte können nun Nachrichten von Nachbarn in Funkreichweite weiterleiten und damit Langstreckenverbindungen ermöglichen [39].

Hypertext Transfer Protocol (HTTP)

Im Gegensatz zu den bereits zuvor vorgestellten Kommunikationsstandards handelt es sich bei HTTP um eine reines Anwendungsprotokoll. Dieses befindet sich im Schichtenmodell in der obersten Schicht.

HTTP ist ein zustandloses Protokoll. Hauptsächlich eingesetzt wird es, um Webseiten aus dem World Wide Web im Browser anzuzeigen. HTTP-Nachrichten haben einen Nachrichtenkopf, den so genannten HTTP-Header, welcher Informationen zu Kodierung, Sprache, Browser und Inhaltstyp enthält. Zusätzlich besitzen HTTP-Nachrichten einen Nachrichtenkörper, Body genannt, der die eigentlich genutzten Daten, wie etwa den HTML-Quelltext, enthält [5].

MQTT

Bei Message Queue Telemetry Transport (MQTT) handelt es sich um ein „leichtgewichtiges, ereignis- und nachrichtenorientiertes Protokoll zur effizienten und asynchronen Kommunikation zwischen Geräten auch über limitierte Netzwerke“ [40].

MQTT befindet sich, ebenso wie HTTP, auf der Anwendungsschicht des Protokollturms. Aufgrund seiner Leichtgewichtigkeit eignet sich MQTT besonders für IoT-Lösungen. Realisiert ist MQTT in einer Publish-/ Subscribe-Architektur.

Im Mittelpunkt der Kommunikation steht der MQTT-Broker. Alle Geräte, die Informationen versenden wollen, schicken diese an den zentralen Broker. Dieser empfängt alle Nachrichten und filtert sie. Geräte, welche diese Nachrichten benötigen und auswerten, habe diese subskribiert. Der Broker sendet diesen dann die Informationen zu. Zusätzlich überwacht der Broker den Status der Verbindungen und reagiert bei Störungen. Die Adressierung beim Senden und Empfangen (publish und subscribe) wird über so genannte Topics realisiert. Hierbei handelt es sich um Zeichenketten, welche eine Art Betreff der Nachricht repräsentieren. Wenn ein Gerät eine Nachricht empfangen möchte, abonniert dieses das entsprechende Topic und wird fortan vom Broker mit den Informationen beliefert [40].

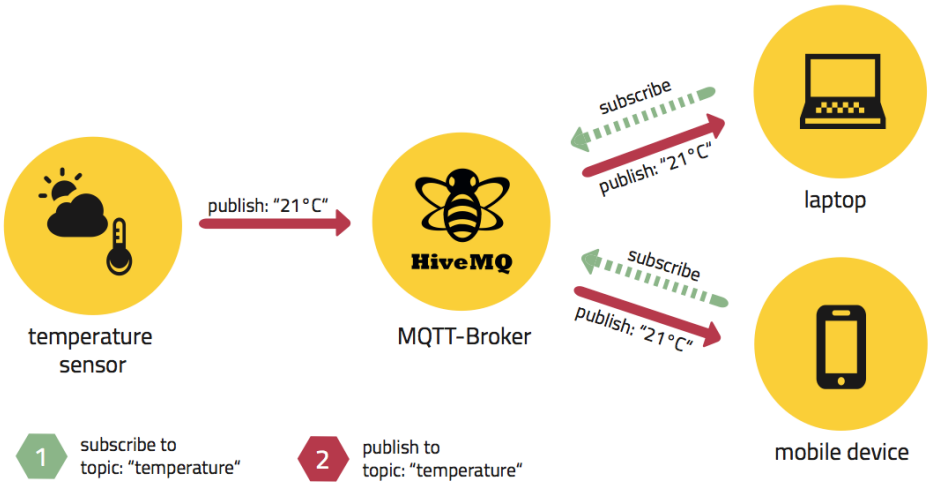


Abbildung 2.6: MQTT Architektur [6]

2.2.6 Cloud Anbindung

Um seine Privatsphäre zu wahren und sich vor Angriffen zu schützen, ist es sinnvoll für Smart-Home-Lösungen, dass sich alle intelligenten Geräte im eigenen, lokalen Netzwerk befinden, geschützt hinter einer Firewall und lokal gesteuert. Alle Daten bleiben zunächst im lokalen Netz. Kai Kreuzer, der Erfinder von OpenHAB, beschreibt dieses Konzept als das „Intranet of Things“ [41]. Wenn man seine Geräte auch von unterwegs steuern möchte oder Cloud-Dienste aus dem Internet genutzt werden sollen, wird ein Gateway zum Internet benötigt. Über dieses ist das lokale, offline Smart-Home-Netzwerk mit dem Internet verbunden. Solche Cloud-Anbindungen sind beispielsweise PaaS-Dienste und cloudbasierte Sprachassistenten.

Platform as a Service (PaaS)

PaaS-Systeme sind webbasierte Entwicklungsplattformen, gehostet von einem Anbieter. Mithilfe dieser kann Code geschrieben werden, es wird die Laufzeitumgebung bereitgestellt und das Management ermöglicht. Ebenso können Daten von IoT-Sensoren grafisch aufbereitet und online jederzeit eingesehen werden. David Linthicum, CEO der Linthicum Group Softwareberatung, beschreibt PaaS so, dass diese dem Nutzer unterschiedliche Werkzeuge bieten, für Bereiche wie das Interface-Design, die Prozesslogik oder die Integration [42].

Ein Beispiel für solch einen Dienst ist der IoT-Plattform-Service ThingSpeak. Dieser erlaubt dem Nutzer das Sammeln, Visualisieren und Analysieren von Live-Datenströmen in der Cloud [43]. Mittels des HTTP-Protokolls werden die Daten über das Internet an den Plattform-Service übertragen.

IFTTT

Bei IFTTT handelt es sich um einen Web-Service, der 2010 ins Leben gerufen wurde. Der Slogan des Dienstes lautet „Put the Internet to work for you“ [44].

Der IFTTT-Dienst ermöglicht Nutzern das Erstellen und Anpassen von aneinandergereihten Bedingungen, so genannten Rezepten, welche bei zuvor definierten Auslösern ausgeführt werden. Solche Auslöser sind beispielsweise Veränderungen in Webdiensten wie Facebook, Twitter, WordPress oder YouTube [45]. Es wird ein bestimmter Auslöser mit einer Aktion verknüpft.

Ein Einsatzbeispiel wäre das Einschalten einer Philips Hue Lampe, sobald man auf Facebook in einem Bild markiert wird oder dass Personen, die einen Kom-

mentar auf der WordPress Seite des Nutzers hinterlassen, eine automatisierte Antwort per Email bekommen [44]. Ebenso könnte die Steuerung des Lichtes mit dem aktuellen Aufenthaltsort des Nutzers verknüpft werden (festgestellt anhand der GPS-Position des Smartphones) und bei Verlassen des Hauses die Lichter automatisch ausgeschaltet werden.

Für die Automatisierungen können Anwender vorgefertigte Rezepte nutzen, welche sie auf der Webseite finden. Ebenso besteht die Möglichkeit, eigene Rezepte zu erstellen, bestehende anzupassen oder welche zu nutzen, die andere IFTTT-Nutzer auf der Webseite bereitgestellt haben.

Für das Erstellen eines IFTTT-Rezeptes muss zuerst der „Trigger Channel“ festgelegt werden. Hierzu wählt man zunächst den Dienst aus, der genutzt werden soll, beispielsweise Instagram. Danach wird das Event gewählt, das das Rezept auslösen soll (z.B. ein neu eingestelltes Foto). Dazu muss dem IFTTT-Dienst der Zugang zum Account des Nutzers gewährt werden. Im nächsten Schritt wird der „Action Channel“ ausgewählt. Hier hat man die Wahl zwischen unterschiedlichen vorprogrammierten Aktionen, die ausgelöst werden, sobald der im Trigger Channel definierte Fall eintritt [44].

Aktuell arbeiten mehr als 360 unterschiedliche Partner mit der IFTTT-Plattform zusammen und stellen Dienste bereit, darunter Amazon Alexa, Nest, Fitbit und Skype [46].

Cloudbasierte Sprachassistenten

Bei Sprachassistenten werden Sprachbefehle aufgenommen und an den Server mit dem Dienst geleitet. Die übermittelten Daten werden ausgewertet und die entsprechende Antwort zurückgeliefert. Die Nutzung von Sprachassistenten zur Steuerung des Smart Homes ist eines der neuesten Anwendungsgebiete und gerade im Hinblick auf die Zukunftsfähigkeit einer Smart Home Lösung ein Punkt, der nicht vernachlässigt werden sollte.

Ein Beispiel für so einen cloudbasierten Sprachassistenten ist Amazon Alexa. Über ein Lautsprecher-System mit Spracherkennung kann der intelligente Sprachassistent Alexa genutzt werden. Über so genannte „Wake-Words“ oder durch Drücken der Aktionstaste wird der Dienst aktiviert und den Worten des Nutzers zugehört. In der Alexa-App auf dem Smartphone können Skills angelegt werden. Diese bieten Funktionalitäten, beispielsweise für die Steuerung von Smart Home Geräten. Nach dem Erfassen des Befehls sucht Alexa die in der App hinterlegten Skills nach dem passenden Befehl ab. Bei einer Übereinstimmung wird dieser ausgeführt und beispielsweise eine Lampe eingeschaltet.

Weitere bekannte cloudbasierte Sprachassistenten sind Google Assistant, Siri oder Microsoft Cortana.

2.2.7 Beispiele für Smart Home Lösungen

Privates Smart Home Szenario

Nico Jurran stellt in der Zeitschrift c't sein privates Smart-Home-Projekt vor. Folgendes Schaubbild gibt einen Überblick über das gesamte Projekt.

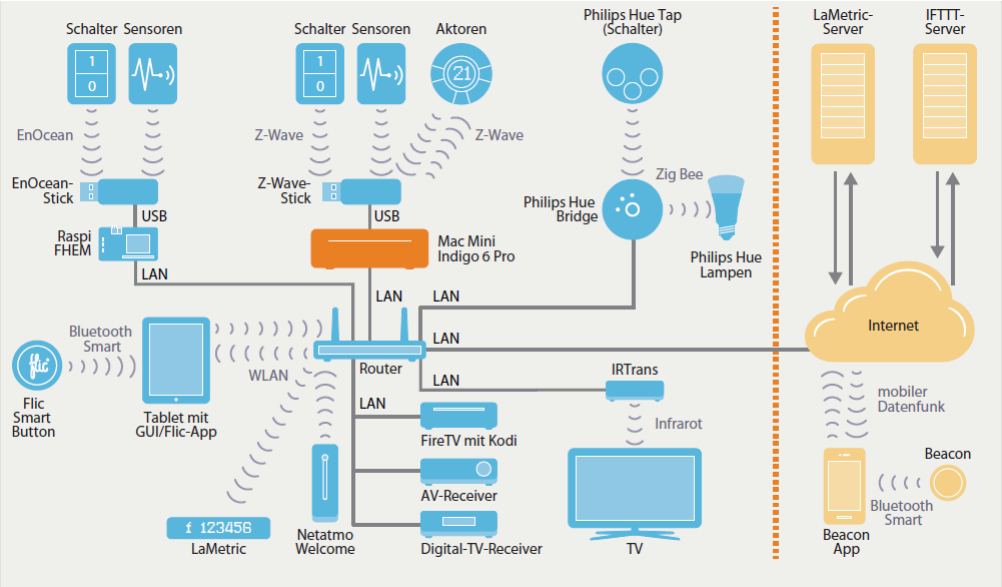


Abbildung 2.7: Smart Home Szenario Architektur [7]

Im Zentrum, als Steuerzentrale, steht ein Mac mini, auf dem die Steuersoftware „Indigo 6 Pro“ läuft. Bei Indigo 6 handelt es sich um eine, käuflich zu erwerbende, Smart-Home-Software. Alternativ könnte hier aber auch auf Open Source Alternativen, wie beispielsweise OpenHAB, zurückgegriffen werden.

Ein an der Steuerzentrale angeschlossener Z-Wave-Stick steuert Z-Wave Geräte und erhält von diesen Sensordaten, welche die Steuersoftware verarbeitet.

Die Phillips Hue Bridge ist über LAN mit der Steuerzentrale verbunden. Die Hue Bridge wiederum steuert mittels ZigBee Phillips Hue Lampen und Schalter.

Für EnOcean Schalter und Sensoren bildet ein Raspberry Pi, an dem ein EnOcean-Stick angeschlossen ist und auf welchem die Steuersoftware FHEM

läuft, die Bridge.

Eine Kamera ist über das WLAN eingebunden. Diese kommuniziert über das Internet mit Hilfe des Webdienstes IFTTT und macht Aufnahmen, wenn Bewegungen wahrgenommen werden.

Als Mediencenter wird Kodi genutzt. Beim Abspielen eines Videos wird automatisch eine eingestellt Lichtszene aktiviert. Ist die Wiedergabe beendet, wird wieder die übliche Beleuchtung eingeschaltet.

Als Bedieneinheit dient ein Tablet, unterwegs kann die Steuerzentrale mittels App kontaktiert werden [7].

Bremen Ambient Assisted Living Lab (BAALL)

Menschen erreichen ein immer höheres Alter. Dadurch wird auch zukünftig die Zahl der Pflegebedürftigen steigen. Das Smart Home bietet hier die Perspektive, dass auch Menschen, die Unterstützung brauchen, noch für eine möglichst lange Zeit selbstständig in ihrer eigenen Wohnung bleiben können. In dem Forschungsprojekt BAALL arbeiten Forscher an einem exemplarischen Smart Home Projekt, welches aufzeigt, wie dies funktionieren soll [8].

Beim BAALL handelt es sich um einen 60 Quadratmeter großen Wohnbereich, der besonders alters- und behindertengerecht sein soll. Dieser befindet sich in den Projekträumen des Forschungsbereichs Cyber-Physical Systems des Deutschen Forschungszentrums für Künstliche Intelligenz in Bremen. Ziel dieses Smart Home Forschungsprojektes ist es, Szenarien für das Wohnen mit altersbedingten physischen sowie kognitiven Einschränkungen zu entwickeln und diese mittels technischer Assistenz zu kompensieren [8].

Ein intelligenter Schrank könnte so beispielsweise Menschen mit Demenz dabei helfen, sich der Jahreszeit entsprechend anzuziehen, wozu sie selbst oft nicht mehr in der Lage sind. Im Winter würde dieser daran erinnern, eine Jacke anzuziehen [47].

Höhenverstellbare Wandschränke und Waschbecken unterstützen das Zusammenleben von Rollstuhlfahrern und Menschen ohne Behinderung. Kameras im Spiegel erfassen die Größe des Gegenübers und führen Waschbecken oder Wandschrank automatisch auf die passende Höhe.
Eine am Nachtschrank angebrachte Wärmebildkamera erkennt Stürze. Liegt eine Person auf dem Boden, werden ein Alarm in Gang gesetzt und programmierte Notfallmechanismen eingeleitet [47].

Abbildung 2.8: Höhenverstellbarer Wandschrank [8]

Weitere Projekte

Hausautomatisierung kann eingesetzt werden, um komplexe Probleme zu lösen, wie im vorherigen Beispiel zur Unterstützung der alternden Bevölkerung. Es kann aber auch dazu genutzt werden, die Vielzahl von kleinen Problemen des Alltags zu lösen.

David Hunt beschreibt in dem Magazin MagPi, wie er mittels Heimautomatisierung eine Lösung für seinen Schlafmangel fand. Er litt darunter, dass der Familienhund, wenn dieser nachts aus dem Haus musste, ihn jede Nacht aufweckte. Um sich zu Helfen, baute David einen Türöffner, welcher auf Bellen reagiert. Ein an einen Raspberry Pi angeschlossenes Mikrofon erkennt das Bellen und der Raspberry aktiviert wiederum einen Auslöser, der die Tür entsperrt. Ein Flaschenzug mit Gegengewicht öffnet die Türe dann sanft [9].

Abbildung 2.9: Automatischer Türöffner [9]

Eric Tsai nahm ein anderes Problem von Hundebesitzern in Angriff. Er ließ

morgens seinen Hund aus dem Haus, damit dieser sein Geschäft verrichten konnte. Hier bestand die Gefahr, dass der Hund wegrennt. Ebenso wusste er nie, wo dieser sein Geschäft verrichtet hatte.

Durch das Anbringen einer Box am Halsband des Hundes löste er diese Probleme. Die Box enthält ein GPS-Modul, einen Neigungssensor und einen RF-Sender. Die GPS Position des Hundes wird permanent an einen Raspberry Pi übertragen, auf dem OpenHAB läuft. Wenn der Hund sich weiter entfernt, als zuvor in OpenHAB festgelegt, erscheint ein Warnsignal. Auf einem Bildschirm wird mittels Google Maps die aktuelle Position des Hundes angezeigt.

Abbildung 2.10: Hunde Tracker [10]

Verrichtet der Hund sein Geschäft, erkennt dies der Neigungssensor an der Haltung des Hundes. Auf dem Bildschirm wird eine Karte angezeigt, in welcher die Position des Hundes markiert wurde [10].

3 Softwarelösungen zur Hausautomatisierung

3.1 Der Begriff Open Source

Der Begriff „Open Source Software“ existiert seit 1998. Die Open Source Initiative (OSI) legte damals bestimmte Kriterien fest, welche erfüllt sein müssen, damit Software als „Open Source” bezeichnet werden darf [48]. Bei der OSI handelt es sich um eine, 1998 gegründete, gemeinnützige Organisation, welche sich der Förderung von Open Source Software widmet. Mitglied der OSI ist unter anderem Bruce Perens, ehemaliger Projektleiter für das freie Debian Projekt, welches eine GNU/Linux-Distribution herausgibt [49].

Laut der Open Source Definition (OSD) der OSI müssen folgende Kennzeichen erfüllt sein, damit eine Software als Open Source bezeichnet werden darf:

- Die Lizenz muss frei weitergegeben werden können. Dabei dürfen keinerlei Lizenzgebühren für die Nutzung anfallen.
- Der Quellcode der Software muss gegeben sein und die Weitergabe muss sowohl für den Quellcode, als auch für den Code in kompilierter Form, zulässig sein.
- Es müssen Veränderungen an der Software erlaubt sein und der veränderte Code unter den gleichen Lizenzbestimmungen weiterverbreitet werden können.
- Die Verbreitung von verändertem Quellcode darf nur dann untersagt werden, wenn eine gesonderte Weitergabe des Quellcodes mit so genannten „Patchdateien“ erlaubt ist. Dies ermöglicht den Entwicklern der Open Source Software, dass eine Trennung zwischen ihrem und dem fremden Code ersichtlich ist. Ebenso kann dann verlangt werden, dass die veränderte Software einen anderen Namen oder eine andere Versionsnummer tragen muss.
- Es darf keine Person und keine Personengruppe von der Nutzung ausgeschlossen werden und auch die Nutzung für bestimmte Einsatzbereiche

darf nicht untersagt werden.

- Die Rechte der Lizenz müssen für jeden gelten, der das Programm erhalten hat und zwar ohne, dass weitere Lizenzen beachtet werden müssen.
- Die Rechte zur Nutzung der Software dürfen nicht davon abhängen, dass das Programm Teil eines bestimmten Softwarepaketes ist.
- Durch die Software darf keine andere Software, die zusammen mit dem lizensierten Programmcode weitergegeben wird, eingeschränkt werden. So darf auch nicht verlangt werden, dass alle Programme, welche auf dem gleichen Medium weitergeben werden, quelloffen sein müssen [48] [50].

Das zentrale Merkmal von Open Source Software ist also die ganzheitliche Gewährung urheberrechtlicher Nutzungsrechte, wodurch das freie Kopieren, Bearbeiten, Untersuchen und Verbreiten ermöglicht wird [48].

3.2 Open Source Plattformen

3.2.1 OpenHAB

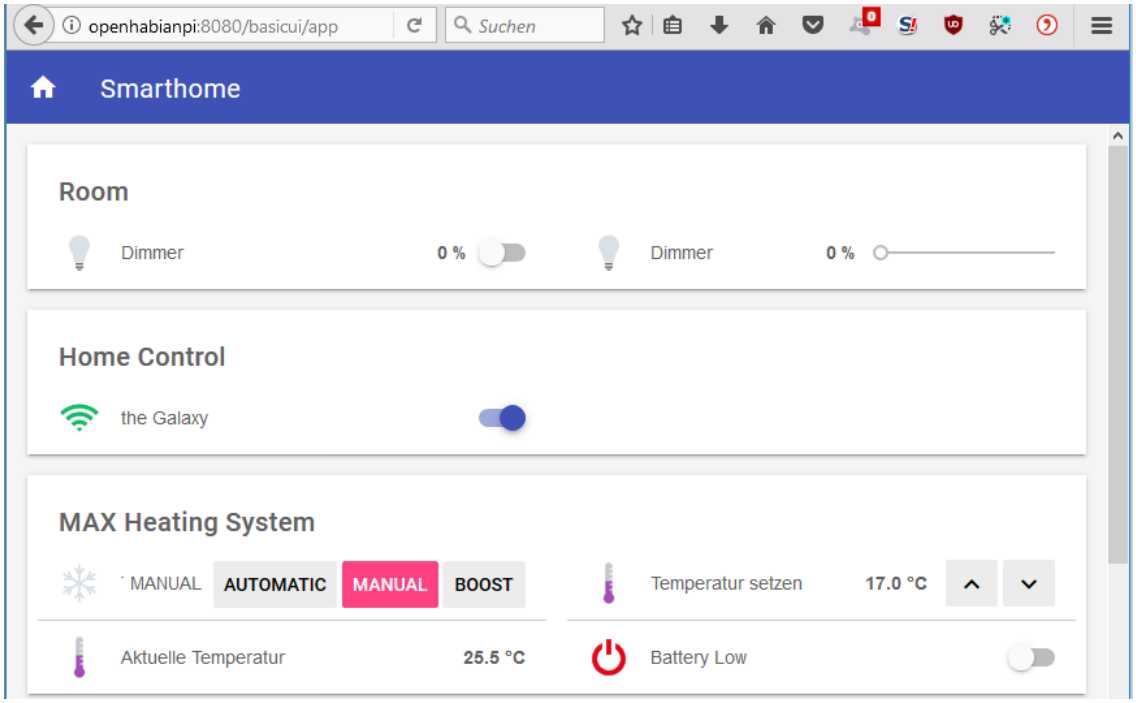


Abbildung 3.1: Beispielhafte Steuerungsoberfläche OpenHAB

Bei OpenHAB handelt es sich um eine Open Source Smart Home Software, die in Java implementiert wurde und auf dem Eclipse SmartHome-Framework basiert. OpenHAB ist komplett herstellerneutral sowie hardware- und protokollunabhängig [51]. Das Projekt mit dem Namen Open Home Automation Bus (OpenHAB), wurde von dem Java-Entwickler Kai Kreuzer im Jahr 2010 ins Leben gerufen. Mit OpenHAB wollte dieser eine robuste, stabile und offene Smart Home Software-Lösung schaffen [52]. Es kann auf jedem Gerät eingesetzt werden, auf dem auch eine Java Virtual Machine genutzt werden kann [51].

Funktionsumfang

OpenHAB-Projekte bestehen aus Things, Bindings, Channels und Items.

Things sind Objekte, die zum System hinzugefügt werden können. Diese können dabei unterschiedliche Funktionalitäten liefern. Bei Things handelt es sich dabei nicht zwingend um Smart-Home-Geräte, es kann sich dabei ebenso um einen Webdienst oder ein Protokoll handeln. Für jedes Thing wird in OpenHAB ein Binding benötigt. Dieses sorgt dafür, dass sich die OpenHAB-Software mit den Things verständigen kann. Vergleichbar wäre ein Binding mit einem Gerätetreiber [51].

Things bieten ihre Funktionalität über so genannte Channels an [51]. So hat eine smarte Glühbirne beispielsweise oft einen Channel zum Einschalten des Lichtes und einen weiteren Channel, um die Leuchtfarbe festzulegen.

Items wiederum werden eingesetzt, um die Channels anzusprechen. Items haben einen aktuellen Status und können Befehle erhalten. Bei Items kann es sich um Schalter, Dimmer, Zahlen oder Zustände handeln [51].

Dem Nutzer werden von OpenHAB bereits 155 Bindings zur Verfügung gestellt. Ebenso können eigene OpenHAB-Bindings als JAR-Datei in OpenHAB eingebunden oder auf diese Weise zwischen Nutzern ausgetauscht werden.

Verbreitung

Bei der Suchanfrage „OpenHAB“ werden bei dem Filehosting-Anbieter GitHub 1.420 Repositories als Ergebnis zurückgeliefert [53].

Mit GitHub-Stars können Nutzer auf GitHub einem Repository folgen, um dieses so später wiederzufinden und bei Veränderungen auf dem aktuellen Stand zu bleiben.

Der binär-Distribution von OpenHAB folgen 376 GitHub-Nutzer. Die openhab1-

addons-Distribution hat hier sogar 3.464 Sterne, die noch jüngere openhab2-addons-Distribution trägt 672 Sterne.

Die Suchmaschine Google liefert bei der Suche nach „OpenHAB“ 417.000 Ergebnisse [54].

Ebenso stellt OpenHAB ein eigenes Forum für Fragen und Probleme bereit. Insgesamt über 18.000 Themen und 179.000 Posts wurden dort getätigt. 17.300 Nutzer werden dort angegeben, 2.800 aktive Nutzer in den letzten sieben Tagen [55].

Die große Anzahl an Ergebnissen bei Google deutet auf eine weite Verbreitung der Softwarelösung hin. Auch die Menge an abgegebenen Beiträgen im offiziellen OpenHAB-Forum deutet auf ein reges Interesse von Smart Home Nutzern und eine hohe Aktualität hin.

Lizenz

OpenHAB ist mit der freien Software-Lizenz Eclipse Public License belegt, welche das Recht auf freie Nutzung, Weitergabe und Veränderung der Software gewährt [56].

Besonderheiten

- OpenHAB stellt eine eigene Linux-Distribution bereit, basierend auf Raspbian, die bereits mit OpenHAB vorkonfiguriert ist. Zusätzlich beinhaltet diese weitere OpenHAB- und hardwarespezifische Voreinstellungen für den Raspberry Pi.

3.2.2 ioBroker

ioBroker ist eine Integrationsplattform für das Internet der Dinge, welche in JavaScript mit Node.js geschrieben wurde und als zentraler Server für Smart Home Lösungen eingesetzt wird.

Es lässt sich auf jeder Hardware und jedem Betriebssystem betreiben, auf dem auch Node.js läuft [57].

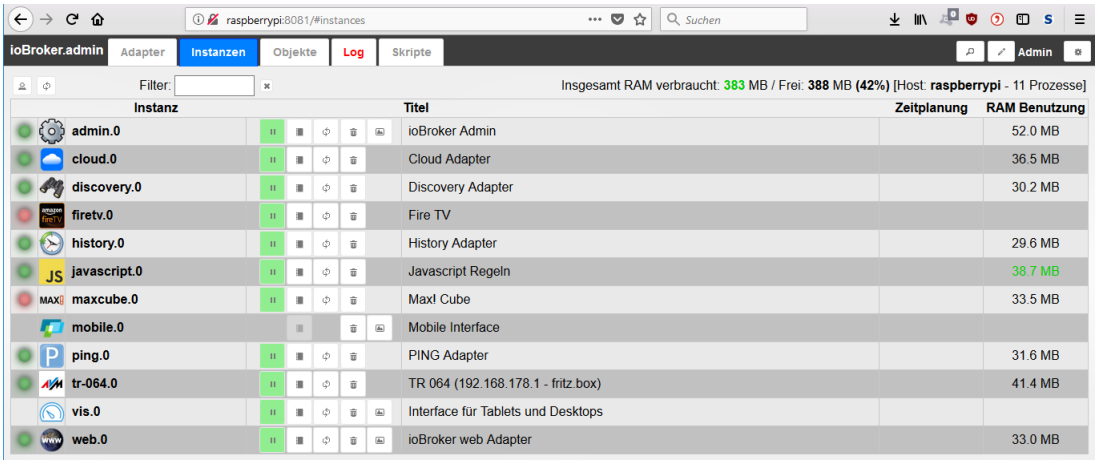


Abbildung 3.2: ioBroker Administrations-Oberfläche

Funktionsumfang

IoBroker Smart Home Lösungen lassen sich unterteilen in Adapter, Instanzen und Objekte.

Über die so genannten „Adapter“ kommuniziert ioBroker mit den unterschiedlichen Smart Home Geräten. Adapter können als eine Art Treiber für ein Gerät, einen Service oder für die Bereitstellung von Daten gesehen werden. Da ioBroker sehr modular aufgebaut ist, müssen auch Admin-Oberfläche, Visualisierung oder auch Module für das Schreiben von eigenen Skripten als Adapter hinzugefügt werden. Durch seinen modularen Aufbau läuft ioBroker sehr stabil und zuverlässig. Der Controller und die einzelnen Adapter laufen als eigenständige Prozesse unter Node.js. Wenn ein Adapter abstürzt, werden andere Prozesse davon nicht beeinflusst.

Nachdem ein Adapter installiert wurde, wird das entsprechende Gerät oder der Dienst als Instanz angezeigt. Die Instanz liefert Status-Informationen und Konfigurationsmöglichkeiten für das Gerät bzw. den genutzten Service. Jeder Adapter hat mindestens eine Instanz. Es können aber auch mehrere Instanzen zu einem Adapter angelegt werden. Dies ermöglicht, mehrere Geräte des gleichen Typs anzusprechen.

Die smarten Geräte bzw. die mittels Adapter hinzugefügten Dienste werden in der ioBroker-Weboberfläche als so genannte „Objekte“ aufgeführt. Die Objekte fungieren als Datenpunkte und geben in einer Ordnerstruktur weitere Zustandsinformationen, wie beispielsweise ob ein Licht an oder aus ist, auf wieviel Prozent die Helligkeit gedimmt wurde oder den Temperaturwert eines

Heizkörperthermostats.

Um logische Abläufe zwischen den Objekten zu definieren muss ein weiterer Adapter installiert werden. In diesem können dann Geräte gruppiert und gemeinsam gesteuert werden. State- und Trigger-Einstellungen werden als Auslöser für die jeweilige Szene festgelegt und so die Abläufe zwischen den Objekten gestartet.

Für eine ansprechende Visualisierung wird der VIS-Adapter bereitgestellt, der ermöglicht eine individuelle Visualisierungs- und Bedienungsoberfläche zu erstellen. Durch eine Auswahl an unterschiedlichen Widgets und Icons, das Einbinden von eigenen Bildern und mittels HTML-Code und unter Zuhilfenahme von CSS, kann das Aussehen den eigenen Wünschen angepasst werden [57][58][59].

Verbreitung

Unter GitHub erhält man 364 Repositories als Antwort auf die Suche nach dem Begriff „ioBroker“.

159 GitHub-Stars wurden an das ioBroker-Repository vergeben [60].

Die Online-Suchmaschine Google liefert ungefähr 161.000 Ergebnisse zu ioBroker [61].

In dem offiziellen ioBroker-Forum wurden insgesamt 98.611 Beiträge zu 8.821 Themen verfasst. Die Mitgliederanzahl beträgt 4.078 [62].

Die Anzahl der Suchergebnisse ist, im Vergleich zu den anderen untersuchten Smart Home Lösungen, verhältnismäßig gering und deutet auf eine entsprechend geringere Verbreitung von ioBroker hin. Speziell die geringe Anzahl an GitHub Repositories könnte darauf hindeuten, dass in Entwicklerkreisen ioBroker noch verhältnismäßig wenig verbreitet ist. Allerdings zeigt die hohe Anzahl an aktuellen Beiträgen im ioBroker-Forum, dass man bei Problemen eine aktive Community an seiner Seite hat.

Lizenz

ioBroker ist unter der Open-Source MIT-Lizenz lizensiert [63].

Besonderheiten

- ioBroker stellt eine fertige Distribution für den Raspberry Pi bereit, auf der bereits eine vollständige ioBroker Installation vorhanden ist.
- ioBroker ist skalierbar. Sollte es an einer Zentralen zu einem Engpass kommen, beispielsweise durch zu wenig noch verfügbare RAM-Kapazitäten, kann eine weitere Zentrale eingebunden werden.
- ioBroker ermöglicht die Nutzung visueller Programmierumgebungen, wie Blockly oder Node-RED, für die Erstellung eigener Skripte. Dafür müssen nur die entsprechenden Adapter installiert werden. Per Drag & Drop lassen sich Skripte nun grafisch zusammenbauen, die automatisch zu JavaScript Code umgewandelt werden [59].

3.2.3 Home Assistant

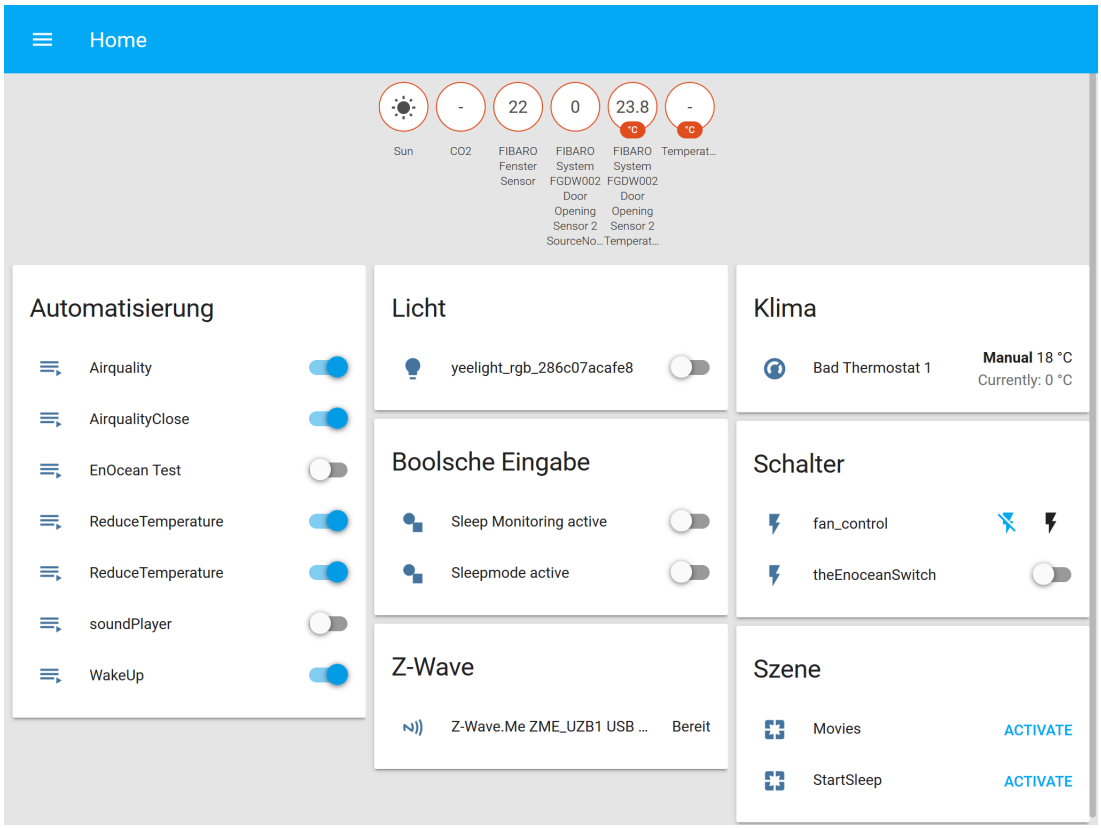


Abbildung 3.3: Beispielhafte Weboberfläche Home Assistant

Bei Home Assistant handelt es sich um eine, auf der Programmiersprache Python basierende, Open Source Home Automation Plattform [64]. Die erste Version von Home Assistant wurde von Paulus Schoutsen entwickelt, der das Ziel verfolgte, sein eigenes Haus zu automatisieren [65].

Funktionsumfang

Zentral bei der Einrichtung der Home Assistant Softwarelösung ist die Markup Language Datei *configuration.yaml*. Diese befindet sich unter dem Pfad ~/.homeassistant/ . Hier findet die grundsätzliche Konfiguration, wie beispielsweise die Festlegung des Standortes, statt. Ebenso werden über diese Datei auch die einzelnen Smart Home Geräte und Web-Services, Components genannt, eingebunden.

Unter https://home-assistant.io/components/ ist online eine Liste der verfügbaren Components zu finden. Hier stehen bereits über 900 unterschiedliche Geräte und Dienste bereit, welche in Home Assistant genutzt werden können. Bei einem Klick auf einen der aufgelisteten Components erhält man den Eintrag angezeigt, der in der configuration.yaml-Datei hinzugefügt werden muss, um den gewünschten Service oder das smarte Gerät in sein Smart Home System zu integrieren.

Für das Einbinden einer Philips Hue Light Bridge muss beispielsweise folgender Code zu der Konfigurationsdatei hinzugefügt werden:

```
light:
- platform: hue
  host: ip-Adresse
```

Hier muss nur noch die entsprechende IP-Adresse der Hue Bridge eingetragen werden und schon wird das Gerät nach einem Neustart in der Weboberfläche von Home Assistant angezeigt und kann genutzt werden.

Automatisierte Abläufe zwischen den einzelnen Components werden in der *automations.yaml* definiert. Eine Automatisierung besteht dabei aus drei Teilen: Trigger (Auslöser), Condition (Bedingung) und Action (Aktion).

- Der Trigger beschreibt das Event, welches den automatisierten Ablauf startet, etwa die Heimkehr des Hausbewohners.
- Bedingungen sind optional. Sie bewirken, dass die Automatisierung nur in bestimmten Fällen ausgeführt wird. Es können mehrere Conditions festgelegt werden. Eine beispielhafte Condition wäre, dass die Sonne bereits untergegangen ist.

- Die Action wird ausgeführt, wenn der Trigger ausgelöst wurde und alle Conditions erfüllt sind. Anhand der vorherigen Beispiele könnte hier die Action sein, dass die Lampen im Haus eingeschaltet werden [66].

Verbreitung

Eine Analyse der Verbreitung von Home Assistant anhand von Suchergebnisse bei Google oder GitHub ist wenig zielführend, da die Kombination der Begriffe Home und Assistant auch für viele andere Fälle genutzt wird. So liefert die Suchanfrage auf Google beispielsweise auch Ergebnisse zu dem neuen Google Home Assistant oder anderen IoT-Hausautomatisierungsprojekten.

Über 10.500 GitHub-Stars für das home-assistant-Repository zeugen von einer großen Zahl an Interessenten für das Open Source Projekt [67].

Im hauseigenen Home Assistant Forum wurden bereits über 175.000 Beiträge zu über 18.000 Themen von 13.900 Nutzern getätigt [68].

Die hohe Anzahl an vergebenen GitHub-Stars deutet auf eine weite Verbreitung von Home Assistant hin. Auch im Forum besteht reges Interesse an der Open Source Lösung.

Lizenz

Home Assistant unterliegt als Open Source Software der Apache License 2.0 Software-Lizenz [69].

Besonderheiten

- Home Assistant stellt mit Hassbian eine eigene Distribution für den Raspberry Pi bereit. Diese basiert auf der Linux-Distribution Raspbian und enthält bereits eine vorkonfigurierte und sofort nutzbare Home Assistant Installation.
- Mit über 900 vorhandenen Components stellt Home Assistant eine immense Anzahl an nutzbaren Geräten und Diensten bereit.

3.2.4 Node-RED

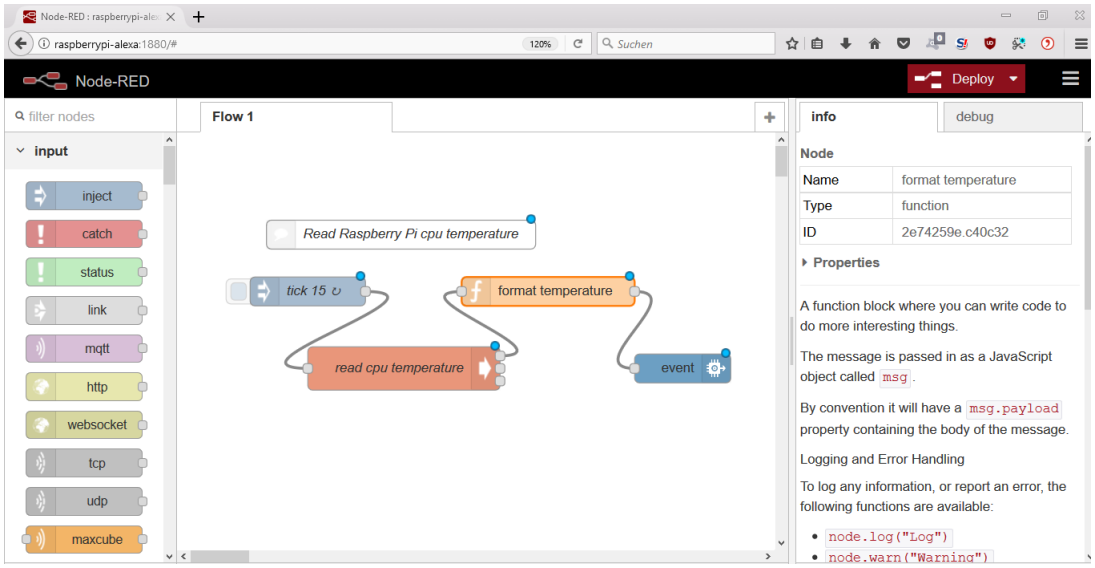


Abbildung 3.4: Browserbasierte Programmierumgebung Node-RED

Node-RED ist eine IoT-Programmierumgebung, mit welcher Hardware-Devices, Programmierschnittstellen (APIs) und Online-Dienste miteinander verknüpft werden können. Entwickelt wurde es im Jahr 2013 als ein Nebenprojekt von Nick O'Leary und Dave Conway-Jones von der IBM Emerging Technology Services Gruppe. Heute ist es Teil der JS Foundation.

Implementiert wurde Node-RED in JavaScript, wobei das Node.js Framework genutzt wurde. Da es sehr leichtgewichtig ist, eignet sich Node-RED ideal für die Nutzung auf kostengünstiger Hardware wie dem Raspberry Pi [70].

Funktionsumfang

Node-RED nutzt eine browserbasierte Programmierumgebung, den so genannten „flow editor". Mit Node-RED erstellte Programme werden „Flows" genannt. Flows bestehen aus vorgefertigten Code-Blöcken, so genannten „Nodes", welche miteinander verbunden werden.

Über den Webbrowser wird der flow editor aufgerufen und in diesem dann die Programme erstellt. Durch Drag & Drop werden im Browser die vorgefertigten Programmblöcke aus einer Palette in den Workspace gezogen und dort miteinander verknüpft.

Eine große Community von Nutzern steht hinter dem Node-RED-Projekt. Die bereits vorhandene Palette an verfügbaren Nodes kann individuell erweitert werden. So besteht die Möglichkeit eigene Nodes in JavaScript zu schreiben oder Code-Blöcke, die andere Nutzer der Community geschrieben haben, nachträglich zu installieren. Im Node-RED package repository befinden sich über 225.000 Module, welche hinzugefügt werden können [71].

Die fertigen Programme werden als JSON-Dateien gespeichert. In einer Online-Bibliothek können die fertigen Flows mit anderen Nutzern ausgetauscht werden.

Verbreitung

GitHub liefert auf die Suchanfrage „Node-RED“ 4.448 Repositories als Suchergebnis [72].

5.116 GitHub-Stars für das Node-RED-Repository zeugen von einer Vielzahl von Nutzern, die bei Node-RED über die neuesten Entwicklungen unterrichtet werden wollen.

Google liefert über 20.500.000 Ergebnisse zu Node-RED [73].

In der Node-RED Online-Bibliothek (flows.nodered.org) werden 718 flows von anderen Nutzern angeboten und die JS Foundation veröffentlich kontinuierlich aktuelle Versionen von Node-RED (zuletzt im Juli 2017).

Speziell bei der Anzahl der Treffer bei Google stellt Node-RED die anderen Plattformen in den Schatten. Auch eine hohe Anzahl an Repositories bei GitHub zeigt, dass Node-RED sehr weit verbreitet ist. Allerdings muss hier auch beachtet werden, dass es sich bei Node-RED, im Gegensatz zu den anderen zu vergleichenden Software-Lösungen, nicht um eine reine Smart Home Lösung, sondern um eine visuelle Programmierumgebung für das Internet der Dinge handelt und es so auch in vielen anderen Bereichen neben dem Smart Home eingesetzt wird.

Lizenz

Node-RED unterliegt der freien Apache License 2.0 Software-Lizenz und ist als Open Source Software frei nutzbar [74].

Besonderheiten

- In einigen Raspberry Pi Distributionen, wie beispielsweise Raspbian Jessie, ist Node-RED bereits vorinstalliert und kann direkt genutzt werden.

4 Smart Home Szenario - Vergleich von vier Open Source Plattformen

Für den Vergleich der Smart Home Plattformen wird im Folgenden ein exemplarisches Smart Home Szenario vorgestellt, das mit jeder der Plattformen umgesetzt wird.

Das Szenario soll realitätsnah den Einsatz von Smart Home Lösungen im privaten Umfeld abbilden und dabei möglichst viele der in 2.2.5 aufgezählten Technologien einsetzen und unterschiedlichste Kriterien, welche heute und zukünftig an ein Smart Home gestellt werden können, umfassen.

Das exemplarische Smart Home Szenario stellt einen **intelligenten Schlafassistenten** für das private Zuhause dar.

Die Umsetzung des Assistenten lässt sich dabei in drei Abschnitte unterteilen: Einen **Einschlafassistenten**, einer **Schlafüberwachung** und einer **Aufweckfunktion**.

Nach dem Start des Einschlafassistenten wird Musik abgespielt und ein Sonnenuntergang simuliert. Nach einer halben Stunde wird die Schlafüberwachung aktiviert.

Für einen erholsamen Schlaf wird die Temperatur des Heizkörperthermostats auf die ideale Schlaftemperatur von 18°C geregelt. Während der Schlafüberwachung wird die Zimmertemperatur sowie der CO2 Wert in der Luft überwacht. Ist einer der Werte zu hoch, wird automatisch gelüftet.

Morgens um 9:00 Uhr wird die Aufweckfunktionalität gestartet. Hierzu wird der Schlafmodus deaktiviert, ein Sonnenaufgang simuliert und Musik abgespielt, um den Nutzer sanft aufzuwecken.

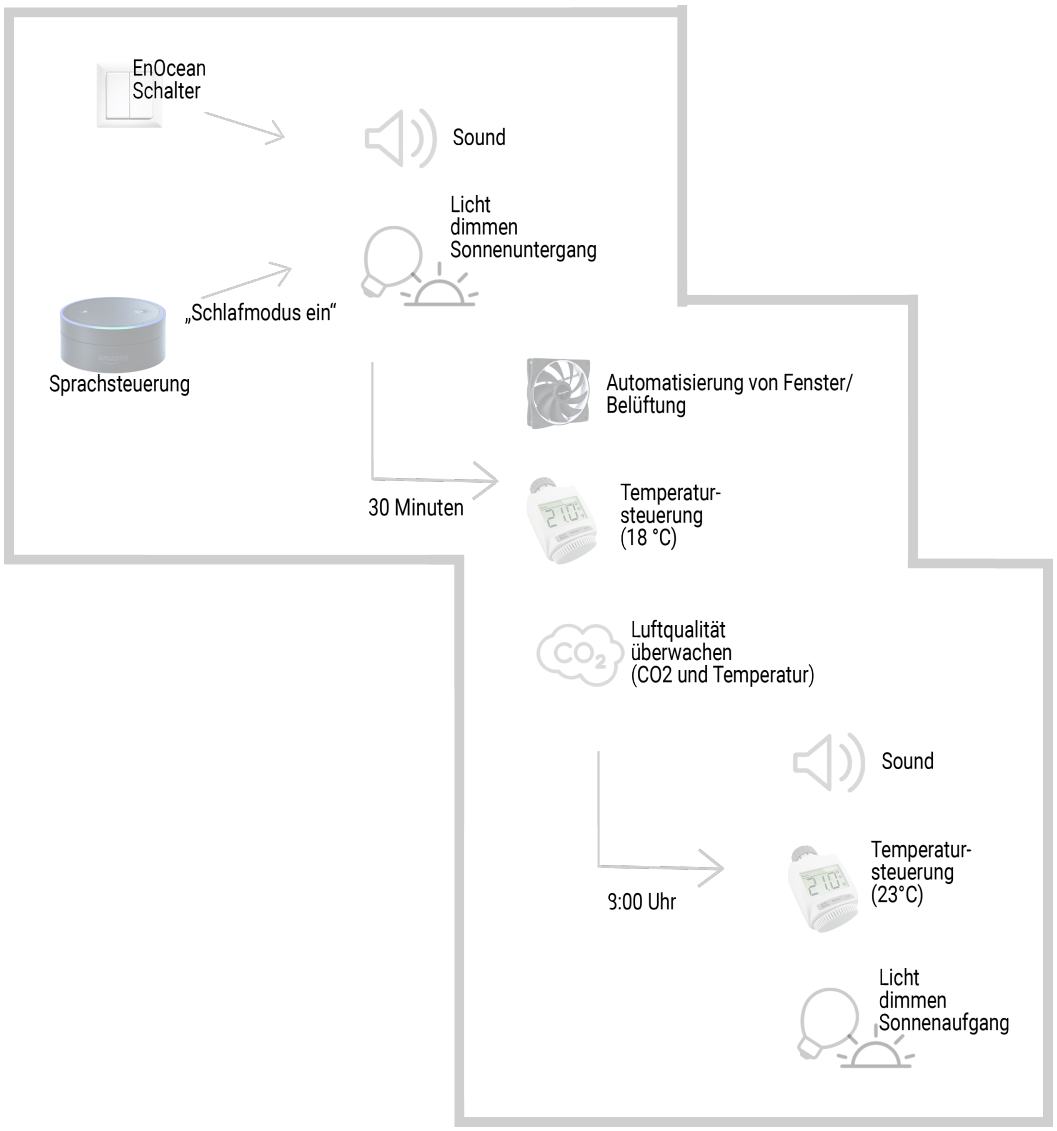


Abbildung 4.1: Ablauf Smart Home Szenario

4.1 Anforderungen

Einschlafassistent

I. Starten des Einschlafassistenten über den Sprachassistenten Amazon Alexa oder

II. Starten des Einschlafassistenten mittels des EnOcean Schaltmoduls

III. Abspielen einer Sounddatei, um den Nutzer beim Einschlafen zu unterstützen

IV. Langsame Verringerung der Helligkeit der smarten LED für die Simulation eines Sonnenuntergangs

V. Starten der Schlafüberwachung nach Ablauf der Einschlaffunktion

Schlafüberwachung

VI. Automatisierte Einstellung des Heizkörperthermostat auf ideale Schlaftemperatur von 18°C

VII. Überwachung von Zimmertemperatur und CO2-Wert in der Luft und automatisierte Belüftung, wenn einer der Werte einen bestimmten Grenzwert überschreitet

VIII. Überprüfung mittels Z-Wave Fenstersensor, ob das Fenster vor dem Start der Belüftung bereits geöffnet ist

Aufweckfunktion

IX. Start der Aufweckfunktion zu zuvor festgelegter Zeit (9:00 Uhr)

X. Deaktivierung der Schlafüberwachung

XI. Setzen der Temperatur des Heizkörper-Thermostats auf 24°C

XII. Langsames Aufhellen der LED für die Simulation des Sonnenaufgangs

XIII. Abspielen einer weiteren Sounddatei

Anwesenheitserkennung

XIV. Ermittlung, ob der Nutzer sich Zuhause befindet anhand der GPS Daten des Smartphones und Aussetzen der Aufweckfunktion, wenn dieser nicht

Zuhause ist

4.1.1 Prioritäten der Anforderungen

Aufgrund der hohen Anzahl an Anforderungen sollen diese anhand ihrer Wichtigkeit für das Smart Home Projekt priorisiert werden. Die Priorisierung geschieht anhand der Wichtigkeit der Funktionalität für das Smart Home an sich. So hat der Einsatz von den typischen Smart Home Kommunikationsprotokollen Z-Wave oder EnOcean eine hohe, das Abspielen ein Sound-Datei, was auf verschiedenste Arten passieren kann, eine eher geringe Priorität.

Ebenso soll die Wichtigkeit der Anforderung für die Sinnhaftigkeit des exemplarischen Smart Home Projekts beachtet werden.

Anforderung	**Priorität**
II.	Hoch
V.	Hoch
VIII.	Hoch
IX.	Hoch
IV.	Mittel
VI.	Mittel
VII.	Mittel
X.	Mittel
XI.	Mittel
XII.	Mittel
I.	Niedrig
III.	Niedrig
XIII.	Niedrig
XIV.	Niedrig

4.2 Vergleichskriterien

Anhand von verschiedenen Vergleichskriterien sollen die vier Smart Home Plattformen untersucht und bewertet werden.

Diese Vergleichskriterien sind:

Installation

Am Kriterium des Installationsaufwandes wird untersucht, wie komplex sich die Installation und die Grundkonfiguration der jeweiligen Smart Home Softwarelösungen darstellt.

Oberfläche

Hier werden Frontend und Backend der Softwarelösungen im Hinblick auf Übersichtlichkeit, logischem Aufbau, Usability und User Experience begutachtet.

Konfiguration

Unter dem Gesichtspunkt der Konfiguration werden zwei unterschiedliche Kriterien herangezogen. Zum einen wird betrachtet, wie viele der unterschiedlichen **Technologien** aus dem Smart Home Szenario sich in die jeweilige Smart Home Plattform integrieren lassen. Als zweites Kriterium wird untersucht, wie groß der Aufwand ist die entsprechenden Geräte und Dienste einzurichten, also die **Einfachheit** der Konfiguration. Konkret überprüft werden die in 2.2.5 vorgestellten Kommunikationsprotokolle EnOcean, Z-Wave, MQTT, ZigBee, HTTP und Bluetooth sowie die Anbindung von Cloudbasierten Sprachassistenten und die Integration der restlichen smarten Geräte des Smart Home Szenarios.

Ein Teil dieser Technologien wird direkt in dem Smart Home Szenario eingesetzt. Aus Kosten- und Zeitgründen wird die Einbindung einiger Kommunikationsstandards allerdings nur anhand der Dokumentation untersucht.

Visualisierung

Für die Bewertung der Visualisierung wird die Darstellung der Bedienoberfläche auf unterschiedlichen Endgeräten umgesetzt. Hierbei werden die Möglichkeit der Erstellung von individuellen Visualisierungen und die Einfachheit der Umsetzung von ansprechenden Bedienoberflächen verglichen.

Erweiterbarkeit

Die Erweiterbarkeit wird untersucht anhand der Möglichkeit, eigene Skripte in die Open Source Plattform zu integrieren und auszuführen und eigenen bzw. externen Code einzubinden, um die Anzahl der nutzbaren Dienste und Geräte zu erweitern.

Automation

Für das Vergleichskriterium der Automation werden automatisierte Abläufe zwischen den Geräten und Diensten implementiert. Untersucht werden soll, ob die Automationen des exemplarischen Smart Home Szenarios mit der jeweiligen Plattform umfassend umgesetzt werden können. Ebenso wird die Einfachheit der Einrichtung der automatisierten Abläufe bewertet.

4.3 Benötigte Hardware

Raspberry Pi

Der Raspberry Pi ist ein Einplatinenrechner, der sich ideal für die Realisierung von Hausautomatisierungsprojekten eignet. Er stellt aufgrund seiner Rechenleistung und seiner Größe von 86 mm x 54 mm x 17 mm den idealen Kompromiss zwischen einem einfachen Mikrocontroller und einem großen Computersystem dar. Die CPU des Raspberry Pi ist in einer ARM-Architektur aufgebaut, welche energiesparend und trotzdem leistungsstark ist [75]. Bei dem neuesten Modell, dem Raspberry Pi 3 Model B, ist die CPU vom Typ ARM Cortex-A53 [76]. Als System on a Chip (SoC) stellt der Raspberry eine CPU, ein GPU sowie die Steuerung für diverse Schnittstellen bereit.

Abbildung 4.2: Raspberry Pi [11]

Ursprünglich entwickelt wurde der Raspberry Pi, um Kindern den Umgang mit dem Computer und das Programmieren näherzubringen. Deshalb wurde bei der Entwicklung des Raspberry Pi der Fokus auf den Preis sowie die Benutzerfreundlichkeit gelegt. Das Raspberry Pi Projekt selbst ist ein Open-Source Projekt. Alle Dokumentationen und Dateien sind frei verfügbar [75].

Im exemplarischen Smart Home Projekt wird der Raspberry Pi als Steuerzentrale eingesetzt. Auf diesem wird die jeweilige Smart Home Plattform installiert und betrieben. Verwendet wird der Raspberry Pi 3 Modell B.

Espressif ESP32 Mikrocontroller und Adafruit CCS811 Luftqualitätssensor

Der ESP32 ist ein kostengünstiger SoC Mikrocontroller. Mit in diesen integriert sind Wireless LAN und Bluetooth [77]. Im Smart Home Projekt wird dieser zusammen mit dem Adafruit CCS811 Luftqualitätssensor genutzt. Dieser kann unter anderem die Umgebungstemperatur sowie den CO2-Wert in der Luft messen. Der Adafruit CCS811-Sensor wird über die GPIO Pins des ESP32 mit diesem verbunden.

Abbildung 4.3: ESP32 [12]

Der Mikrocontroller übermittelt im exemplarischen Smart Home Projekt die Daten, die er von dem Luftqualitätssensor erhält, an den Raspberry Pi bzw. die eingesetzte Smart Home Plattform. Dafür wird das IoT-Protokoll MQTT eingesetzt.

EnOcean-Gateway und Mehrkanalschaltmodul

EnOcean USB 300 ist ein USB-Stick, der als Gateway den Raspberry Pi mit den batterielosen Funkgeräten von EnOcean verbindet. Er bietet bidirektionale Funkkommunikation mittels EnOcean-Protokoll und empfängt alle Radio-Telegramme, die von EnOcean Geräten in der näheren Umgebung versendet werden [78]. Jeder EnOcean Sensor versieht seine Telegramme mit seiner persönlichen Geräte-ID, über welche die Telegramme unterschieden und gefiltert werden können.

Abbildung 4.4: PTM210 [13]

Der PTM 210 ist ein Funksensor-Schalt-Modul von EnOcean, das die batterielose (Fern-) Steuerung von Geräten ermöglicht. Die, für die Über-

tragung des Signals benötigte Energie, wird durch einen eingebauten elektrodynamischen Generator erzeugt und ein RF Telegramm an das EnOcean Gateway übermittelt.

Z-Wave Controller und Kontaktsensor

Der Z-Wave USB-Stick verbindet den Raspberry Pi, bzw. die genutzte Smart Home Software, mit dem Z-Wave Netzwerk. Über diesen lässt sich der Computer als Z-Wave Controller verwenden und somit Z-Wave Geräte steuern. Ebenso empfängt er Nachrichten von Z-Wave Sensoren, welche von der Smart-Home-Software ausgelesen und weiterverarbeitet werden können.

Beim dem genutzten Tür- und Fenstersensor handelt es sich um einen batteriebetriebenen Magnet-Kontakt-Sensor, der mit Z-Wave kompatibel ist. Mit dem Sensor kann der Zustand (geöffnet oder geschlossen) von Türen und Fenstern überwacht werden. Das Sensor-Gerät besteht aus zwei Einzelteilen. Ein Teil wird dabei an das Fenster angebracht, der andere an den Rahmen. Liegen sich die beiden Komponenten gegenüber, besteht für den eingebauten Magnet-Sensor ein Kontakt und das Fenster wird als geschlossen gemeldet. Zusätzlich misst der Sensor die Temperatur am Installationsort. Der aktuelle Status des Kontaktsensors und die Temperatur werden mittels Z-Wave Kommunikationsprotokoll an die Z-Wave Steuerzentrale übermittelt.

Abbildung 4.5: Kontaktsensor [14]

In dem exemplarischen Projekt werden der Z-Wave.Me USB Smart Home Stick (ZMEEUZB1) und der Fibaro Tür- und Fenstersensor (FIBEFGDW-002-1) eingesetzt.

Heizkörperthermostat mit Cube LAN Gateway

Das intelligente Heizkörperthermostat von eQ-3 ermöglicht die Steuerung der Heizung über das Internet bzw. das lokale WLAN. Der so genannte MAX! Cube fungiert als Schnittstelle zwischen den Funkthermostaten und der Heizungssteuerung und gibt die Steuerdaten an das Thermostat weiter. Die Kommunikation mit dem MAX! Cube findet über das WiFi-Netzwerk statt.

Abbildung 4.6: MAX! [15]

Smarte LED-Lampe

Abbildung 4.7: LED [16]

Für die Lichtsteuerung im Smart Home wird die Xiaomi Yeelight Smart LED verwendet. Verbunden über das WLAN lässt sich die LED Lampe mittels einer Smartphone-App ansteuern und dabei Helligkeit, Farbe sowie Farbtemperatur festlegen. In der Geräte-Dokumentation wird die API ausführlich beschrieben und so kann die Glühbirne beispielsweise auch über die Kommandozeile mit dem Kommandozeilenwerkzeug Netcat direkt über das lokale WLAN angesteuert werden. Ein beispielhafter Befehl für die Festlegung der Helligkeit der Lampe über die Linux-Kommandozeile könnte folgendermaßen aussehen:

```
1 $ echo -ne '{ "id": 1, "method":"set_bright", "params":
    [1","smooth",500]} \r\n' | nc -w1 192.12.2.3 55443
```

433 MHz Funk-Sendemodul und Funksteckdose

Abbildung 4.8: Sendemodul [17]

Mit einem 433 MHz Sendemodul können Hochfrequenz-Funkübertragungen im ISM Band durchgeführt werden. Das Sendemodul moduliert die Daten und überträgt sie anschließend an den Empfänger. Als Empfänger wird im Smart Home Szenario eine Funksteckdose genutzt, welche eigentlich über einen Handsender gesteuert wird. Da der Handsender sein Signal mit 433 MHz überträgt, kann

die Funksteckdosen auch mit dem 433 MHz Sendemodul angesprochen werden. An die Funksteckdose ist ein Ventilator angeschlossen. Dieser soll die Belüftung im Smart Home Szenario simulieren.

Amazon Alexa

Um den Schlafassistenten zu aktivieren, wird der cloudbasierte Sprachassisten Amazon Alexa genutzt. Eine genauere Beschreibung von diesem ist in 2.2.6 zu finden.

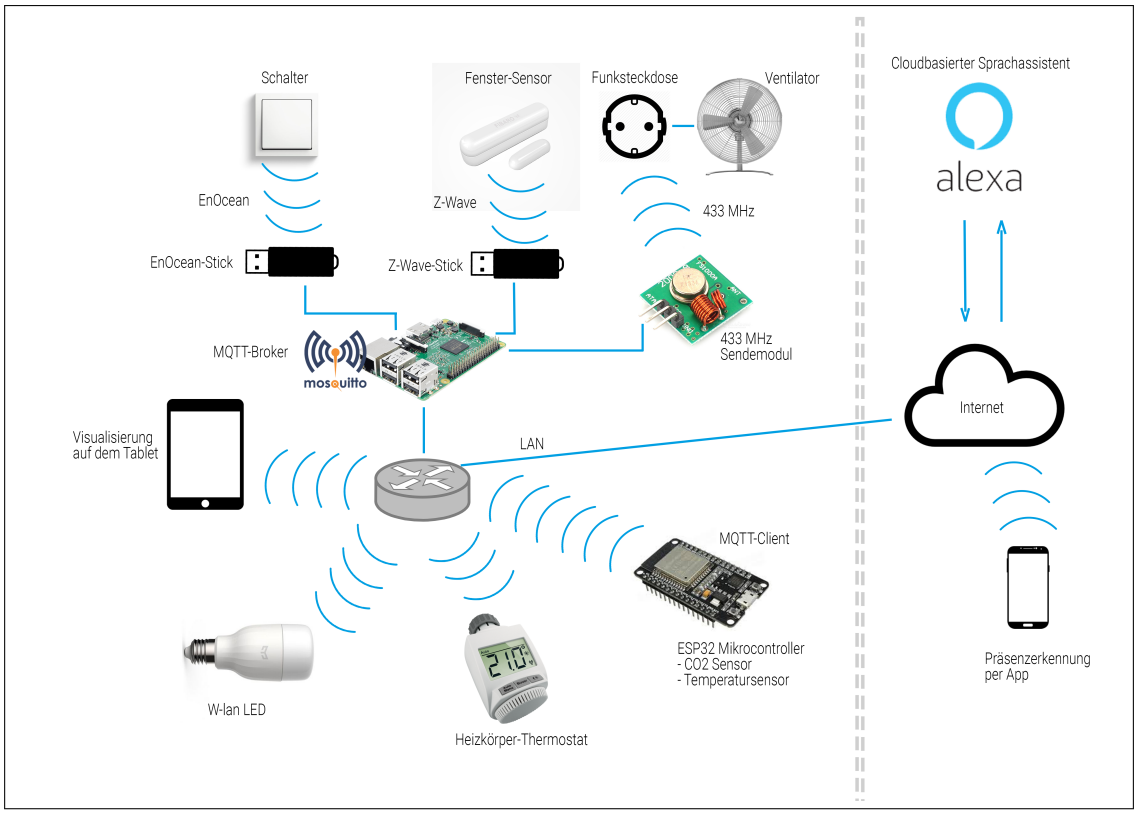


Abbildung 4.9: Benötigte Geräte Smart Home Projekt

4.4 Werkzeuge - genutzte Software

PuTTY

Für die Arbeit mit dem Raspberry Pi empfiehlt es sich auf diesen von einem anderen PC über eine remote Session zuzugreifen. Über das Netzwerkprotokoll SSH kann eine verschlüsselte Netzwerkverbindung zu dem Raspberry aufgebaut und so, entfernt von einem anderen PC, die Kommandozeile des Raspberry bedient werden. Dies bringt den Vorteil, dass man von seinem normalen, leistungsstärkeren Arbeits-PC aus mit dem Raspberry arbeiten kann, ohne diesen an einen Monitor und Eingabegeräte anzuschließen. Im Gegensatz zu UNIX-Maschinen ist bei Windows kein SSH-Client verfügbar. Um das Protokoll zu nutzen, muss deshalb zusätzliche Software installiert werden.

Eine der verbreitetsten Softwarelösungen stellt PuTTY dar. Bei PuTTY handelt es sich um ein Client-Programm für SSH, Telnet und Remote login. Dieses stellt die Oberfläche zur Verfügung, um textuelle Befehle an den Raspberry Pi zu senden [79].

Samba

Der Samba-Service ermöglicht den Datenaustausch in Netzwerken. So kann Samba beispielsweise als Server dienen und ein Datenträger, wie die Festplatte, mit anderen Rechnern im Netzwerk geteilt werden. Auf diese Weise können einfach Daten zwischen dem Arbeits-PC und dem Raspberry Pi ausgetauscht, dieser als File-Server genutzt und auch Systemdateien und Scripte auf dem PC in dem präferierten Editor bearbeitet werden [80].

Mosquitto

Mosquitto ist eine leichtgewichtige Open Source Software, welche einen Broker für das Kommunikationsprotokoll MQTT bereitstellt und sich gut auf dem Raspberry Pi betreiben lässt.

Um die aktuellste Version von Mosquitto auf dem Raspberry Pi zu installieren muss mit folgendem Befehl zunächst der aktuelle Signaturschlüssel des Repository Packages geholt,

```
$ wget http://repo.mosquitto.org/debian/
  mosquitto-repo.gpg.key
$ apt-key add mosquitto-repo.gpg.key
```

dem Paketmanager verfügbar gemacht und anschließend installiert werden.

```
$ cd /etc/apt/sources.list.d/
$ sudo wget http://repo.mosquitto.org/
  debian/mosquitto-stretch.list
$ sudo apt-get update
```

Danach können Mosquitto Broker, Client und die dazu gehörende Python-Library installiert werden

```
$ sudo apt-get install mosquitto mosquitto-clients
  python-mosquitto
```

OwnTracks App

Für die Anwesenheitserkennung wird die App OwnTracks verwendet. OwnTracks ermöglicht den Nutzern, ihre GPS-Position mittels des Smartphones zu dokumentieren. Diese Daten werden an einen MQTT-Broker weitergegeben, der diese wiederum an einen beliebigen MQTT-Client übermitteln kann. Durch einen Abgleich der empfangenen Daten mit festgelegten Koordinaten des Zuhauses kann festgestellt werden, ob der Nutzer gerade anwesend ist.

raspberry-remote und wiringpi

Für die Ansteuerung von Funksteckdosen im Smart Home Szenario bietet sich das Programm „raspberry-remote“ an. Das 433 MHz Sendemodul wird dazu mit den GPIO Pins des Raspberry Pi verbunden.

Die benötigten Services „wiringpi“ und „raspberry-remote“ können per git auf den Raspberry Pi geladen werden.

```
$ sudo apt-get install git git-core
$ git clone git://git.drogon.net/wiringPi
$ cd
$ git clone git://github.com
  /xkonni/raspberry-remote.git
$ cd raspberry-remote
$ make send
$ cd
$ sudo mv raspberry-remote /opt/raspberry-remote
```

Das „send“-Tool benötigt drei Parameter: Den so genannten Hauscode der Funksteckdose, einstellbar mit fünf Schaltern an der Rückseite der Steckdose

(jeweils der Wert 1 oder 0), die Nummer der Dose für die eindeutige Identifizierung, wenn mehrere Steckdosen genutzt werden (hier einer der Schalter A bis E) und das Kommando in Form einer „1“ für „an“ oder einer „0“ für „aus“

Abbildung 4.10: Einstellungen Funksteckdose

Ein Befehl für das Einschalten der Steckdose könnte nun folgendermaßen aussehen:

```
1 $ sudo ./send 11111 1 1
```

4.5 Realisierung exemplarisches Projekt

4.5.1 OpenHAB

Installation

OpenHAB stellt mehrere Möglichkeiten für die Installation auf einem Raspberry Pi bereit. Für den Raspberry empfiehlt sich die Installation in dem, auf Debian basierenden und für den Raspberry Pi optimierten, Betriebssystem Raspbian. Vor der Installation von OpenHAB muss bereits Java, in der empfohlenen Version, installiert sein. Deshalb muss die installierte Java Version zuvor überprüft und gegebenenfalls angepasst werden. Die für OpenHAB empfohlene Version ist Java 8, Revision 101. Java 9 wird bisher (Stand: Dezember 2017) noch nicht unterstützt.

Die anspruchsvollste Installations-Variante ist die manuelle Installation. Diese wird in der OpenHAB-Dokumentation nur Nutzern empfohlen „welche wissen was sie tun“ [81].

Diese lässt sich allerdings in wenigen Schritten durchführen:

- Anlegen des Linux-Users *openhab*

- Zuweisung von Rechten
- Herunterladen und Entpacken der neuesten OpenHAB Version
- Ausführen von OpenHAB

Allerdings müssen an diesem Punkt noch weitere Konfigurationen für eine komfortable Nutzung von OpenHAB durchgeführt werden. Diese sind beispielsweise das Einrichten des Autostart-Services, damit OpenHAB bei einem Systemstart automatisch aktiviert wird oder auch das automatische Erstellen von Backups und das selbstständige Updaten des Systems.

Da Raspbian auf Debian basiert, bietet sich als alternative Installationsmethode die Installation über den Paketmanager an. Diese Installation lässt sich mit wenigen Befehlen über die Kommandozeile durchführen.

Die komfortabelste Installationsmethode von OpenHAB ist über die vorkonfigurierte Raspbian Version „openHABian“. Diese wurde von Thomas Dietrich erstellt und beinhaltet bereits die neueste Version von OpenHAB und weitere notwendige Konfigurationen und Installationen, wie die empfohlene Java-Version und die Autostart-Konfiguration. Die Distribution openHABian wird als .iso-Image-Datei auf die SD-Karte des Raspberry Pi gespielt. So ist OpenHAB direkt beim ersten Starten des Systems nutzbar [81] [82].

Oberfläche

Nach der Installation kann die Oberfläche von OpenHAB im Browser unter der URL `http://ip-des-raspberry:8080` aufgerufen werden.

Hier stehen verschiedene User-Interfaces für die Konfiguration und Steuerung zur Auswahl: Paper UI, Basic UI, Classic UI, HABmin und Habpanel

1) PaperUI

Die PaperUI ist die Web-Konfigurationsoberfläche. Hier lassen sich ein großer Teil der OpenHAB-Grundkonfigurationen erledigen, OpenHAB-Instanzen einrichten und diese konfigurieren. Am linken Bildschirmrand der PaperUI befindet sich eine Navigation.

Unter dem Navigationspunkt **Control** werden die bereits verbundenen Devices angezeigt. Unter dem Reiter **Add-ons** können OpenHAB-Bindings per Mausklick installiert und deinstalliert werden. Sobald das Binding installiert ist und neue Geräte gefunden wurden, werden diese in der **Inbox** angezeigt und können von dort aus konfiguriert werden.

Unter **Configuration** können bei neueren Bindings die Channels direkt den

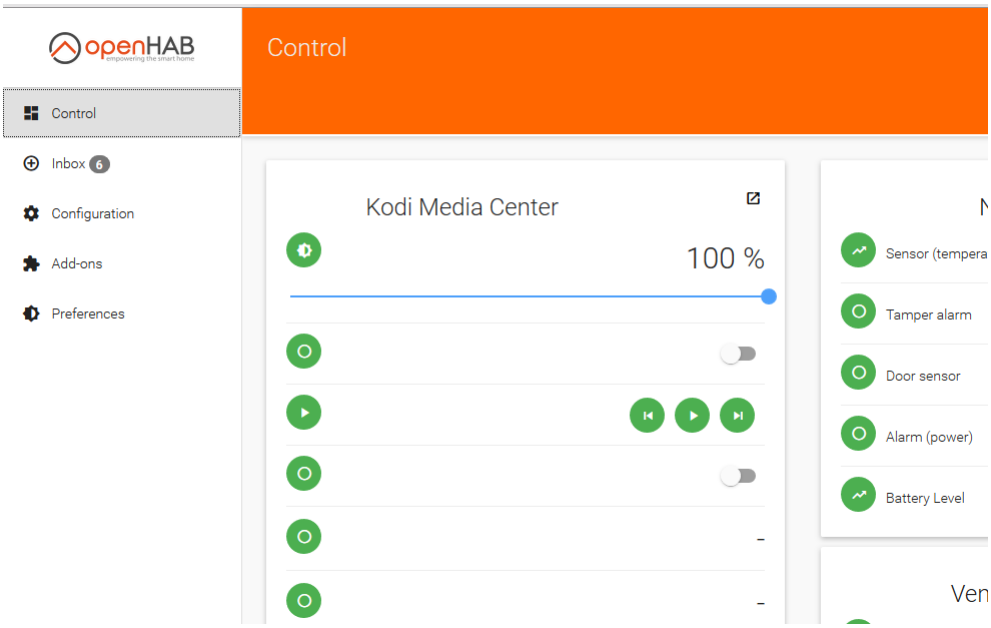


Abbildung 4.11: OpenHAB PaperUI

Items zugeordnet werden und müssen nicht mehr, wie noch bei der älteren Version von OpenHAB, textuell bestimmt werden.

Für die meisten Bindings kann in der PaperUI allerdings nur die Grundkonfiguration der Things durchgeführt werden. Es ist bei der Mehrheit der Bindings weiterhin erforderlich, die notwendigen Items textuell anzulegen und sie auf diese Weise mit den einzelnen Channels zu verknüpfen [83].

2) Basic UI

Unter Basic UI befindet sich eine Bedienoberfläche für das Smart Home. Diese muss allerdings zuerst textuell definiert werden [83].

3) Classic UI

Die Classic UI ist eine weitere Bedienoberfläche, nur in anderem Design. Sie orientiert sich an dem früheren Design von iOS-Geräten [83].

4) HABmin

Das HABmin-Interface kann nachträglich über die PaperUI zur Auswahl hinzugefügt werden. Es handelt sich dabei um eine Administrations-Interface-Alternative zu PaperUI. Hier sollen dem Nutzer noch weitere Konfigurationsmöglichkeiten in einem alternativen Design gegeben werden [83].

5) HABpanel

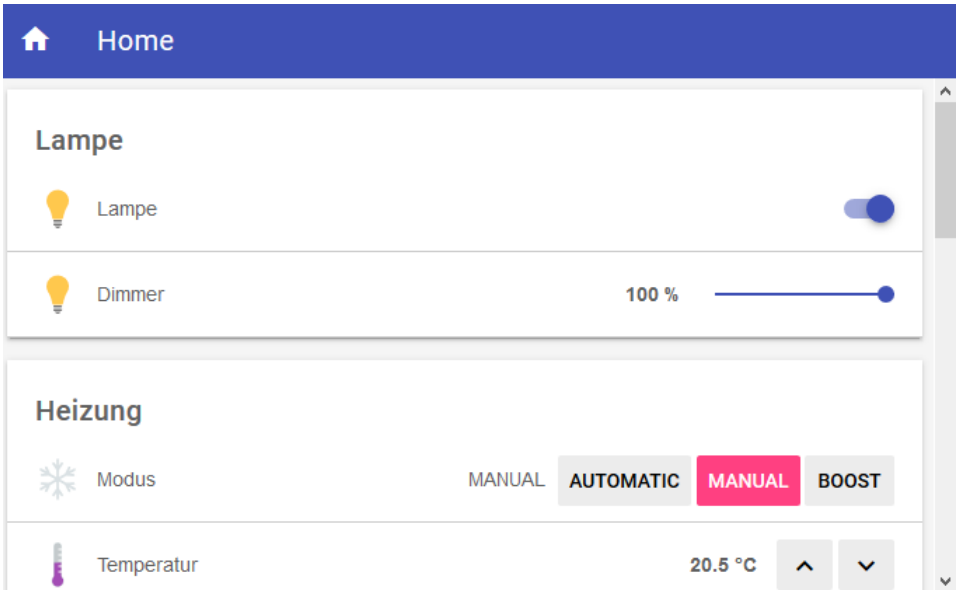


Abbildung 4.12: OpenHAB BasicUI

HABPanel muss, ebenso wie HABmin, nachträglich über die Paper UI installiert werden. Die UI gibt dem Nutzer eine weitere Möglichkeit sich, neben Basic UI und Classic UI, eine Bedienoberfläche einzurichten [83].

Konfiguration

Neue Bindings werden in der Konfigurationsoberfläche PaperUI unter dem Navigationspunkt *Add-ons* installiert.

Nachdem das Binding installiert wurde, muss ein Item angelegt werden. Bei neueren Bindings können Items direkt in der Paper UI erstellt werden. Bei Bindings, die noch vor der OpenHAB 2.x Version erstellt wurden, geschieht das Anlegen der Items über einen Texteditor. Grundsätzlich können auch Items für die neueren Bindings textuell erstellt werden. Items haben einen Status und werden eingesetzt, um die Channel, also die Funktionalitäten der Things, anzusprechen. Ein Beispiel wäre hier ein Item, das mit dem Dimmer-Channel einer LED-Leuchte verknüpft ist. Über dieses Item kann die Helligkeit der Lampe bestimmt werden.

Um ein Item anzulegen, muss zunächst eine Textdatei auf dem Raspberry Pi unter `/etc/openhab2/items` angelegt werden. Der Name der Textdatei kann dabei frei gewählt werden, sie muss allerdings die Dateiendung `.items` haben.

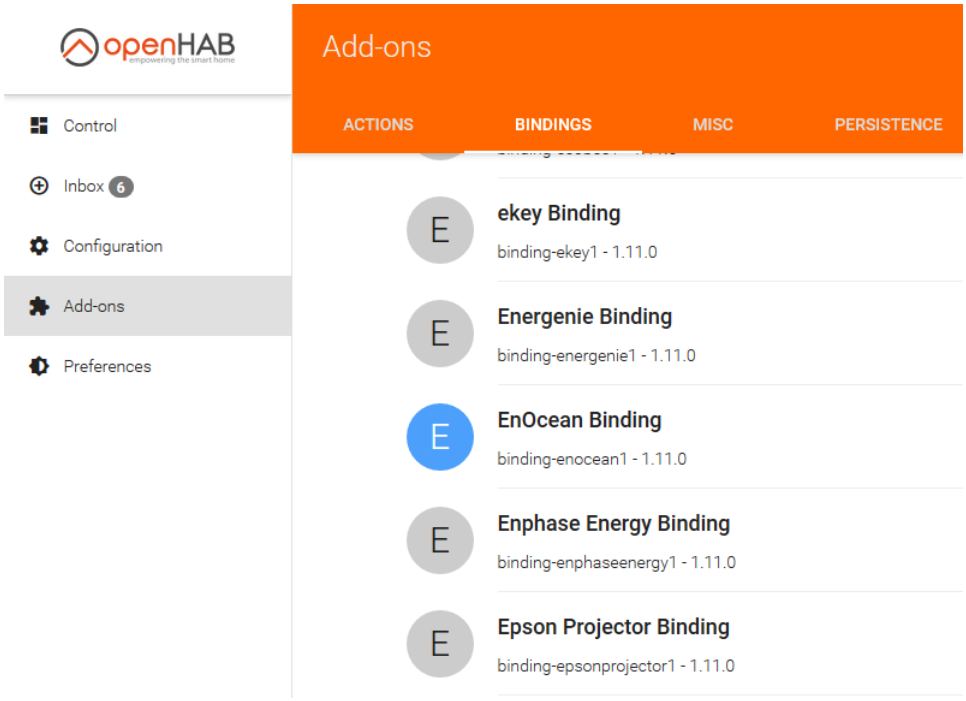


Abbildung 4.13: OpenHAB Add-ons

Für die Definition von Items ist folgende Syntax vorgeschrieben:

```
itemtype itemname "label [stateformat]" <icon>
(group1, ...) ["tag1", "tag2", ...] {bindingconfig}
```

Hierbei sind „itemtype“ und „itemname“ Pflichtangaben. Die restlichen Felder sind optional und binding-spezifisch.

Der Item-Typ definiert die Art und Weise, wie der Status eines Things gespeichert wird. Ein Item vom Typ Color speichert etwa Farbinformation als RGB, ein String-Item beinhaltet Text.

Bei der Installation eines Bindings über die PaperUI wird zusätzlich unter dem Pfad `/etc/openhab2/services` eine Konfigurationsdatei angelegt, sofern diese benötigt wird. In dieser müssen weitere Angaben für die Nutzung des Bindings gemacht werden, wie z.B. die Angabe des Pfades des genutzten USB-Portes oder Längen- und Breitengrade für die Lokalisierung des Standortes.

Für die Konfiguration des **EnOcean**-Gateways muss nach der Installation des entsprechenden Bindings über die PaperUI die automatisch erstellte `enocean.cfg`-Datei angepasst werden. In diese muss eingetragen werden, an welchem USB-Port das Gateway angeschlossen ist.

```
serialPort= /dev/ttyUSB0
```

Mit folgender Syntax wird anschließend ein Item angelegt:

```
enocean="{id=<id_enocean_device> [,eep = <EEP_name>]
[,channel = <channel>][, parameter = <parameter>]}"
```

ID und EEP-Name sind auf dem Gerät, das mit dem EnOcean-Gateway kommunizieren soll, aufgedruckt und dienen der Identifizierung des Gerätes. Der Channel gibt bei einem Schaltmodul an, welcher Knopf gedrückt wurde.

Die fertige Konfiguration sieht dann folgendermaßen aus:

```
Switch SchlafSWEnOcean "Sleep-Mode Activation Button"
{ enocean="{id=XX:XX:XX:XX, eep=F6:02:02, channel=B}"}
```

Die Konfiguration des **Z-Wave**-Gateways kann nach der Installation des Bindings direkt in der PaperUI unter *Configuration* > *Things* durchgeführt werden. Hier muss lediglich der genutzte USB-Port angegeben werden.

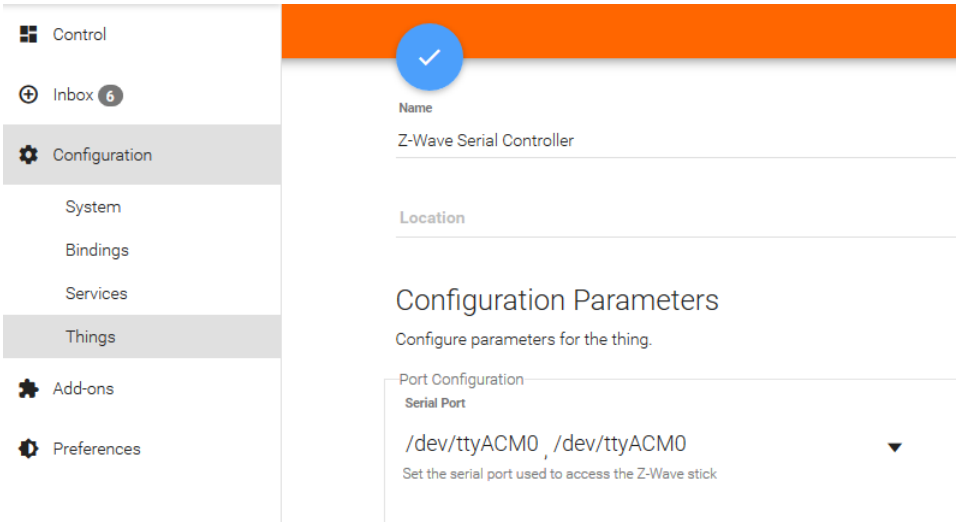


Abbildung 4.14: Z-Wave Konfiguration

In der Inbox der Paper UI befindet sich im rechten oberen Eck eine Schaltfläche mit einem kleinen Antennen-Symbol. Über diese kann das Pairing mit Z-Wave Geräten gestartet werden.

Ist dieses aktiviert, hat man 30 Sekunden Zeit den Z-Wave-Sensor in den Pairing-Mode zu versetzen (bei dem Fensterkontakt-Sensor wird dieser durch fünfmaliges Drücken eines Knopfes auf der Rückseite gestartet). Unter *Configuration* > *Things* werden die gefundenen Geräte angezeigt.

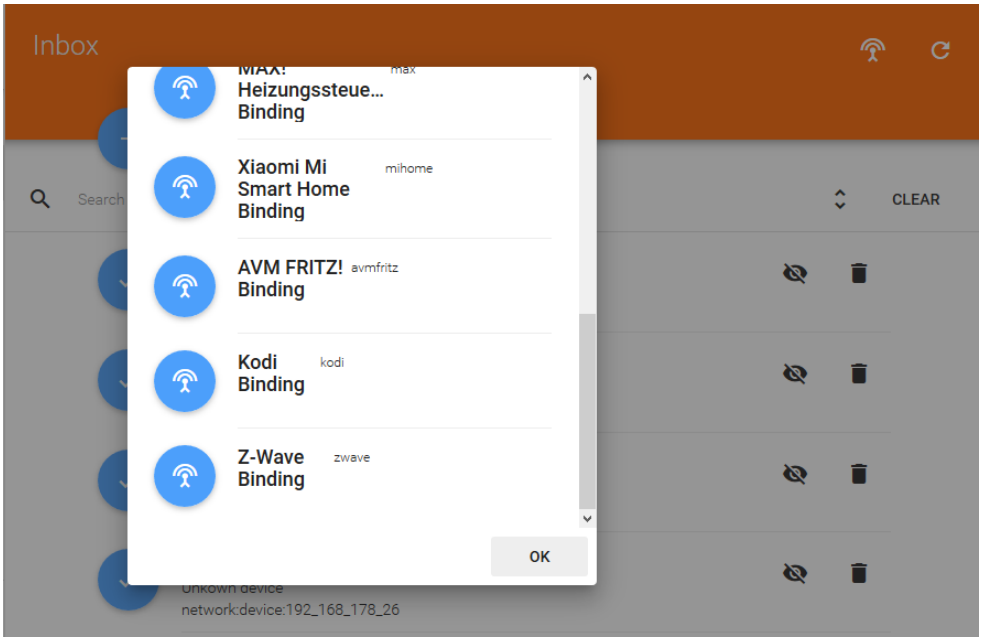


Abbildung 4.15: Z-Wave Pairing starten

Anschließend muss noch ein Item für den Sensor angelegt werden.

```
Contact   window_state     "Fenster: "
{zwave="2:command=SENSOR_BINARY,respond_to_basic=TRUE"}
```

Das Einbinden des **MAX! Cube Heizkörperthermostats** kann vollständig über die PaperUI durchgeführt werden. Die Konfiguration wird in der Inbox durch die Eingabe der Seriennummer des MAX! Cubes und der RF Adresse des Thermostats verrichtet. Anschließend müssen noch Items für die Anzeige der aktuellen und das Setzen der gewünschten Temperatur angelegt werden:

```
Number maxActual  "Aktuelle Temperatur  [%.1f C]"(gMAX)
    {channel="max:thermostat:KEXXXX:actual_temp"}
Number maxSetTemp "Temperatur [%.1f C]" (gMAX)
    {channel="max:thermostat:KEXXXX:set_temp"}
```

Für die **Yeelight LED** stellt OpenHAB kein Binding zur Verfügung. Allerdings hat sich ein OpenHAB-Nutzer diesem Problem angenommen und ein selbstentwickeltes Binding auf GitHub zur Verfügung gestellt. Da dieses Binding allerdings nicht zum standardmäßigen Repertoire von OpenHAB gehört, wird auf dieses genauer unter Erweiterbarkeit eingegangen.

Für die Nutzung des MQTT-Bindings müssen in der `mqtt.cfg`-Datei die URL und der Port des MQTT Brokers angegeben werden.

```
mqtt:broker.url=tcp://localhost:1883
```

Für die Verarbeitung der Daten wird ein Item vom Typ „Number“ angelegt und das Topic, das abonniert werden soll, angegeben.

```
Number airquality "CO2 Wert [%.1f]"{mqtt=
"<[broker:room/airquality/eco2:state:default]"}

Number temperaturemqtt "Temperatur [%.1f]"{mqtt=
"<[broker:room/airquality/temperature:state:default]"}
```

Für die Nutzung des **cloudbasierten Sprachassistenten** Amazon Alexa muss zunächst das Cloud Connector-Binding installiert werden. Dieses ermöglicht die Verbindung des sich im lokalen Netzwerk befindenden OpenHAB-Services mit dem Internet. Für die Nutzung des Cloud Connectors muss zusätzlich unter `https://myopenhab.org/login` ein Account erstellt werden.

Um die Items über Amazon Alexa ansprechen zu können, müssen diese mit Tags gekennzeichnet werden. Folgende Tags können genutzt werden:

- [„Switchabel“] - Für Items vom Typ Switch, Dimmer und Color
- [„Lightning“] - Speziell für Lampen-Items vom Typ Switch, Dimmer und Color
- [„CurrentTemperature“] - Für Geräte, welche die aktuelle Temperatur als Item vom Typ Number zurückliefern

Ein Eintrag für einen Schalter könnte dann folgendermaßen aussehen:

```
Switch DemoSW "Demo Switch" [ "Switchable" ]
```

Anschließend muss in der Alexa-App auf dem Smartphone der OpenHAB-Skill heruntergeladen werden.

In diesem muss man sich mit dem zuvor eingerichteten OpenHAB-Cloud-Account anmelden. Anschließend kann in der App nach neuen Geräten gesucht werden und die mit den Tags gekennzeichneten Items werden nun als neue Geräte in der App angezeigt und können per Sprachbefehl gesteuert werden.

Für die Anwesenheitserkennung mittels **OwnTracks** stellt OpenHAB ein passendes Binding bereit. Ist dieses über die PaperUI installiert worden, müssen in der Konfigurationsdatei `mqttitude.cfg` noch die Koordinaten des Zuhauses angegeben werden:

```
home.lat=xxx.xxxxx
home.lon=xxx.xxxxx
geofence=100
```

Das Item wird, unter Angabe der Broker-ID und des Topics, als Schalter angelegt. Dieser zeigt anhand seines Zustandes an, ob der Nutzer Zuhause ist.

```
Switch  Presence_PhoneMqtt   "PG@Home"
{mqttitude="broker:owntracks/user1/phone" }
```

Für die, im exemplarischen Projekt nicht verwendeten, Kommunikationsprotokolle **ZigBee, HTTP** und **Bluetooth** gibt es in OpenHAB entsprechende Bindings, was die Vermutung nahelegt, dass diese auf ähnlich einfache Art und Weise in das System integriert werden können.

Visualisierung

Um Items anzusprechen bzw. ihre Werte in der Bedienoberfläche (bspw. Basic UI) anzeigen zu lassen, muss eine so genannte „Sitemap“ angelegt werden. Diese definiert das Layout der Bedienoberfläche. Hierfür muss zunächst, wie bereits für das Anlegen der Items, eine Textdatei unter `/etc/openhab2/sitemaps` angelegt werden. Auch hier ist die Dateiendung wichtig, welche `.sitemap` lauten muss (z.B. home.sitemap)

Eine Sitemap-Datei beginnt immer mit dem Wort „sitemap“, gefolgt von dem Namen der Datei.

Eine Sitemap für die Bedienung einer Lampe (Schalter und Dimmer) könnte dann folgendermaßen aussehen:

```
sitemap home  label="Home"{
    Frame label="Lampe" {
        Switch item=Room_Light_Switch icon="light"
        Slider item=Room_Light
    }
}
```

Für die Darstellung stehen zusätzlich verschiedene Icons zur Verfügung, wie „light“ oder „wallswitch“, welche dann in der Bedienoberfläche neben dem Eintrag angezeigt werden.

Ebenso kann die HABPanel-UI nachträglich installiert und mit dieser eine Bedienoberfläche in Kacheloptik per Drag & Drop zusammengestellt werden.

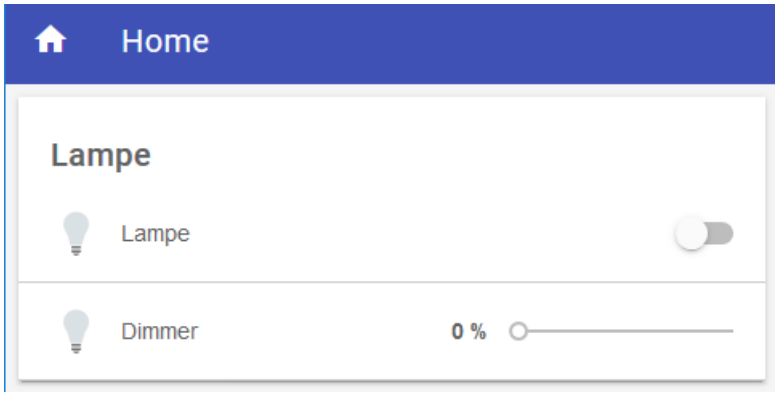


Abbildung 4.16: OpenHAB Sitemap Beispiel

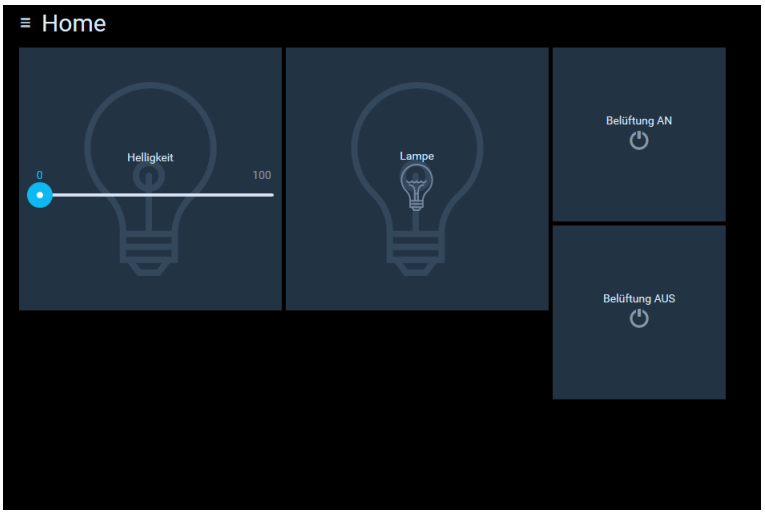


Abbildung 4.17: OpenHAB HABPanel

Erweiterbarkeit

OpenHAB bietet mit dem Exec-Binding die Möglichkeit, Kommandozeilen-Befehle über seine Bedienoberfläche oder automatisiert auszuführen. So können aus OpenHAB heraus Programme, die auf dem Raspberry Pi installiert sind, gestartet und auch verschiedene Scripte, wie Shell-, PHP- oder Python-Scripte, verwendet werden. Über das Exec-Binding soll im Smart Home Szenario der *raspberry-remote*-Service genutzt werden, um eine Funksteckdose anzusprechen.

Um den entsprechenden Befehl über das Exec-Binding absenden zu können, muss zunächst dem User openhab erlaubt werden, den Befehl für das Sende-Tool mit Admin-Rechten auszuführen.

Hierzu wird die Sudoers-Datei mit folgendem Befehl aufgerufen:

```
$ visudo
```

und im dadurch geöffneten Sudoers-File folgende Zeile bei den User-Privilegien hinzugefügt:

```
1 openhab ALL=NOPASSWD: /opt/raspberry-remote/send*
```

Der auszuführende Befehl kann, unter Angabe des Pfades zu dem Service, in der PaperUI unter *configuration* > *things* direkt im Exec-Binding angegeben

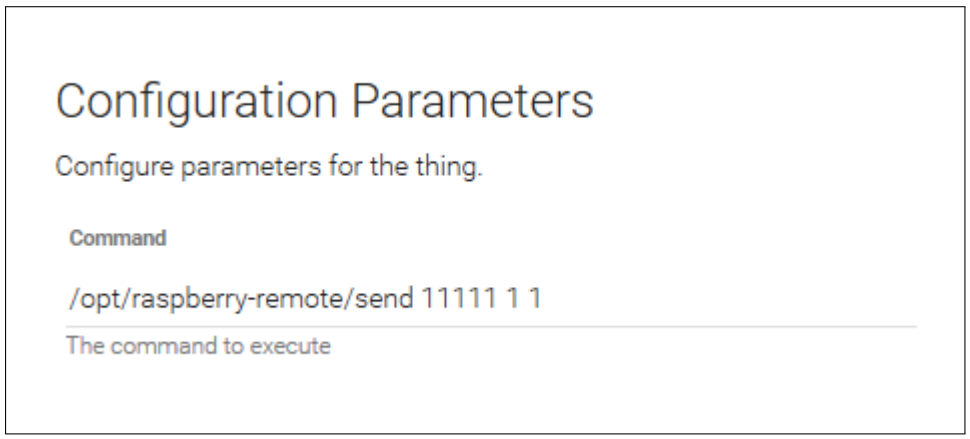


Abbildung 4.18: Exec-Binding Konfiguration

und anschließend das entsprechende Item angelegte werden.

```
1 Switch wallplug_on "Belueftung"
      channel="exec:command:eaaadbbe:run"
```

Ebenso bietet OpenHAB die Möglichkeit, eigene Bindings mittels Java zu schreiben oder externen Code als JAR-Datei einzubinden.

Um das externe Binding zu nutzen, muss dieses als JAR-Datei in OpenHABs Addons-Ordner unter `/usr/share/openhab2/addons` abgelegt werden. Anschließend lässt sich das neue Binding in der PaperUI nutzen. Diese Möglichkeit wird im Smart Home Szenario genutzt, um ein Yeelight-Binding aus GitHub in OpenHAB zu integrieren.

Automation

Für die Automation, also die Einrichtung von Interaktionen und intelligenten Abläufen zwischen den in OpenHAB integrierten Geräten, werden in OpenHAB so genannte „Rules“ genutzt.

Für die Erstellung von Rules muss zunächst eine Textdatei angelegt werden. Diese muss unter dem Pfad `/etc/openhab2/rules` erstellt werden und die Dateiendung `.rules` besitzen. Jede „rules“-Datei kann mehrere Regeln beinhalten. Jede Regel innerhalb einer Datei kann auf die restlichen Regeln in der selben Datei zugreifen und mit diesen Variablen austauschen.

Für das Schreiben von Regeln stellt OpenHAB den Eclipse SmartHome Designer zur Verfügung, der zusätzlich installiert werden kann. Dieser bietet farbliche Syntax-Hervorhebung, validiert den geschriebenen Code und unterstützt den Nutzer so bei der Erstellung einer neuen Regel.

Eine Regel in OpenHAB muss folgende Syntax aufweisen:

```
rule "<RULE_NAME>"
when
    <CONDITION> [or <CONDITION2> [or ...]]
then
    <SCRIPT_BLOCK>
end
```

Die Syntax der Regeln basiert auf der Xtend und ist sehr ähnlich mit der Java-Syntax.

Um den Status eines Items innerhalb einer Regel zu verändern, kann entweder der Befehl `MyItem.postUpdate(new state)`, der den Status des Items ändert ohne weitere Aktionen miteinzuschließen oder alternativ die Anweisung `MyItem.sendCommand(new state)`, welche den Status des Items ändert und mögliche weitere Trigger bzw. Aktionen in Gang setzt, genutzt werden.

Der Status eines Items kann mit dem Befehl `MyItem.state` abgefragt werden [84].

Für das Erstellen der Automationen ist es zusätzlich hilfreich, sich die Log-Files anzeigen zu lassen. Dies geschieht in der Karaf-Konsole. Über die Kommandozeile des Raspberry Pi wird dazu zunächst eine SSH Verbindung über den Port 8101 hergestellt.

```
$ ssh -p 8101 openhab@localhost
```

Das Passwort, nach dem im nächsten Schritt gefragt wird, lautet „habopen“. Nachdem die Verbindung über SSH aufgebaut ist können mit dem Befehl

```
$ log:tail
```

die geloggten Informationen angezeigt werden.

Umsetzung Szenario

Anforderung I und II:

Für das Starten des Szenarios über den Sprachassistent Alexa oder alternativ das EnOcean-Schaltmodul, müssen zunächst zwei Items angelegt werden. Der Trigger überwacht die beiden Items. Wird einer der Trigger ausgelöst, wird die Automation ausgeführt.

```
when
    Item DemoSW changed from OFF to ON or
    Item EnOceanButton changed from OFF to ON
```

Anforderung III:

Wenn der Trigger ausgelöst wird, startet die Musik. Diese kann als Sounddatei unter `/etc/openhab2/sounds` in OpenHAB abgelegt und mit dem Befehl `playSound(SoundName)` abgespielt werden.

```
then
    playSound("sleep.mp3")
```

Anforderung IV:

Um den Sonnenuntergang zu simulieren, muss das Licht langsam gedimmt werden. Da die YAML-Syntax von OpenHAB der von Java sehr ähnlich ist kann hier mit einer Schleife gearbeitet werden, die den Helligkeitswert immer um einen bestimmten Wert verringert, bis die Lampe aus ist. Mit dem Befehl `Thread::sleep(zeitInMS)` kann festgelegt werden, wie lange zwischen einem Dimm-Schritt gewartet werden soll.

```
while(percent >=  0) {
    Room_Light.sendCommand(percent)
    percent --
    Thread::sleep(5000)
}
```

Anforderung V:

Nach dem Sonnenuntergang wird der Status des SleepNow-Items, welches als Trigger für die Schlafüberwachung dient, geupdatet.

Die komplette Regel für den Einschlaf-Assistenten sieht folgendermaßen aus:

```
rule "Fall_Asleep"
```

```
    when
        Item DemoSW changed from OFF to ON or
        Item EnOceanButton changed from OFF to ON
    then
        playSound("sleep.mp3")

        var percent = 100
        while(percent >=  0) {
            Room_Light.sendCommand(percent)
            percent --
        Thread::sleep(5000)
        }
        SleepNowSW.postUpdate(ON)
end
```

Anforderung VI:

Mit dem Start der Schlafüberwachung wird die Temperatur des Heizkörperthermostats auf 18 Grad Celsius gesetzt, wozu der „sendCommand“-Befehl genutzt wird.

```
when
    Item SleepNowSW changed from OFF to ON
then
    maxSetTemp.sendCommand(18)
end
```

Anforderung VII und VIII:

Für die Überwachung der Zimmertemperatur und des CO2-Wertes werden zwei separate Regeln verwendet. Trigger ist jeweils die Änderung eines der Werte. Wenn dies geschieht, wird über eine If-Abfrage überprüft, ob der Schlafmodus aktiviert und der Grenzwert überschritten wurde. Wenn dies der Fall ist, wird die Funksteckdose aktiviert und damit die Belüftung gestartet.

```
rule "Airquality_Monitoring_On"

when
    Item airquality changed
then
    if(airquality.state > 600
            && SleepNowSW.state == ON )
        wallplug.sendCommand(ON)
end
```

Vor dem Starten der Belüftung wird überprüft, ob das Fenster bereits geöffnet ist und die Belüftung deshalb nicht benötigt wird. Hierzu wird der Status des Z-Wave-Sensors überprüft.

```
if(temperaturemqtt.state > 22
      && zwave_node2_sensor_door.state == CLOSED )
```

Anforderung IX - XII:

Für den Start der Aufweckfunktion wird ein „Cron“-Trigger verwendet. Dieser nutzt das „Cron-Daemon“-Tool, das für zeitbasierte Ausführungen auf Unix-Betriebssystemen dient.

```
when
    Time cron "0 0 9 1/1 * ? *"
```

Zu gegebener Uhrzeit wird die Schlafüberwachung durch Aktualisierung des Item-Status ausgeschaltet. Die Temperatur des Thermostats wird auf 24 Grad Celsius gesetzt und die Helligkeit der LED langsam erhöht. Zeitgleich wird eine weitere Sounddatei abgespielt.

```
rule "wakeup_rule"

when
    Time cron "0 0 9 1/1 * ? *"
then
    SleepNowSW.postUpdate(OFF)
    maxSetTemp.sendCommand(24)
    var percent = 0
        while(percent <= 100) {
            Room_Light.sendCommand(percent)
            percent = percent + 10
            Thread::sleep(1000)
}

playSound("wakeSound.mp3")

end
```

Anforderung XIV:

Aufgrund zeitlicher Engpässe konnte die letzte Anforderung, das Aussetzen der Aufweckfunktion, wenn der Nutzer nicht zu Hause ist, nicht mit in die Automation aufgenommen werden.

Das Einbinden der Lokalisierung mittels OwnTracks-App in die Smart Home Plattform funktionierte aber bereits problemlos. Dies legt die Vermutung nahe, dass auch diese Anforderung bei mehr Zeitreserven hätte umgesetzt werden können.

4.5.2 ioBroker

Installation

Voraussetzung für die Installation von ioBroker ist das Vorhandensein von Node.js auf dem Raspberry Pi. Node.js darf dabei nicht in der aktuellsten Version 9.5.0 (Stand Februar 2018) sein, da eine zuverlässige Funktionsweise von ioBroker mit dieser nicht garantiert werden kann. Empfohlen wird eine Node.js Version 6.x.

Vor der Installation muss die auf dem System vorhandene Node.js-Version mit dem Befehl `node-v` überprüft und Node.js gegebenenfalls neu installiert werden.

ioBroker lässt sich anschließend mit wenigen Befehlen über die Kommandozeile des Raspberry Pi einrichten [85].

Alternativ stellt ioBroker eine eigene Raspbian-Distribution bereit, auf der ioBroker bereits installiert und konfiguriert ist.

Oberfläche

Die Weboberfläche von ioBroker wird über `http://ip-des-raspberry:8081` erreicht. Unter dieser Adresse öffnet sich der sogenannte „Adapter Admin“. Dieser dient der Bedienung der ioBroker-Installation. Über diese Weboberfläche findet u.a. die Installation neuer Adapter, der Zugriff auf die Objektübersicht und die Zustände der Objekte, der Zugriff auf das Logfile und die Verwaltung der Hosts statt [86].

Das User Interface besteht aus mehreren Reitern. Es ist nach dem ioBroker-Prinzip in Adapter, Instanzen und Objekte unterteilt. Weitere Reiter können über das Bleistift Icon, welches sich rechts oben befindet, nach Installation der entsprechenden Adapter, hinzugefügt werden.

Unter dem Reiter **Adapter** werden die zur Installation zur Verfügung stehenden Adapter, sowie weitere Informationen, wie die aktuelle Versionsnummer und die neueste verfügbare Adapter-Version, angezeigt.

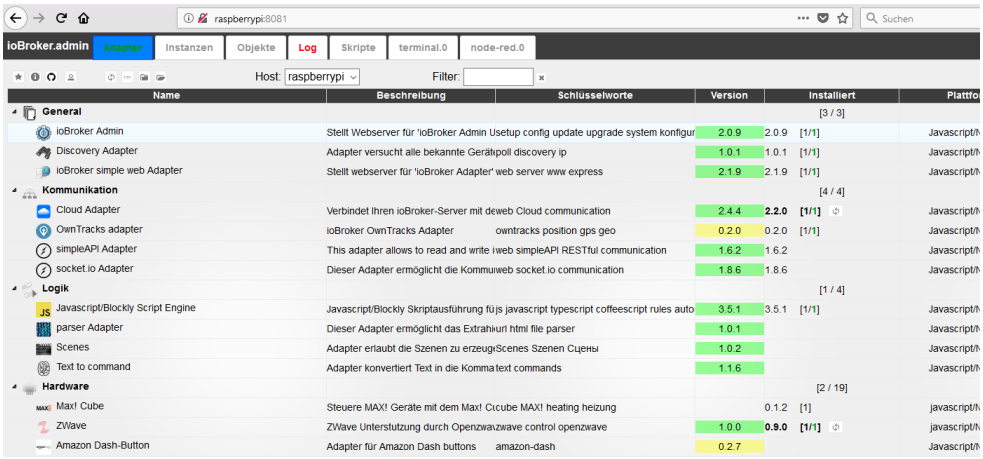


Abbildung 4.19: ioBroker Web-UI

Unter dem Reiter **Instanzen** werden die bereits installierten Adapter als Instanzen gelistet. Diese können hier konfiguriert werden.

Bei **Objekte** befinden sich alle verwalteten Objekte. In einer Ordnerstruktur wird zu jeder der Instanzen ein eigener Ordner angelegt und die Datenpunkte, beispielsweise mit dem MaxCube-Adapter verbundene Thermostate, hierarchisch in dem jeweiligen Ordner aufgelistet [86].

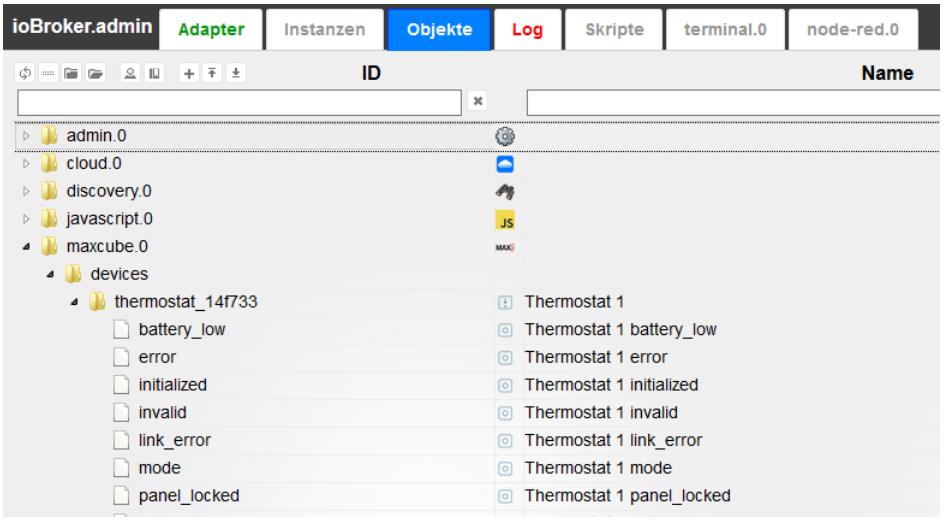


Abbildung 4.20: ioBroker Objekte

Konfiguration

Um die smarten Geräte und die verschiedenen Dienste in ioBroker nutzen zu können, müssen sogenannte „Adapter“ installiert werden. Damit diese Adapter verwendet werden können, müssen Instanzen von diesen erzeugt und konfiguriert werden. In der Weboberfläche unter dem Reiter Adapter können diese über ein Plus-Icon per Mausklick installiert werden. Wenn der Adapter installiert ist, wird automatisch eine Instanz von diesem erzeugt, welche unter dem Reiter Instanzen aufgelistet wird. Im Reiter Instanzen findet die Konfiguration des Dienstes oder des Gerätes statt. Über einen Klick auf den Adapter wird das Konfigurationsfenster geöffnet, wo weitere Angaben, wie der genutzte USB-Port oder die IP-Adresse des Gerätes angegeben werden müssen. Nach der Konfiguration kann der Adapter über ein Play-Icon gestartet werden. Eine Anzeige links neben dem Adapter zeigt farblich (grün, gelb, rot) den Verbindungsstatus an [87].

Für die Nutzung von **EnOcean**-Geräten wird mindestens Node.js in der Version 7 benötigt. Da aber die Nutzung von ioBroker mit Node.js 7.x oder höher aufgrund von Fehlern noch nicht empfohlen wird, ist der EnOcean-Adapter noch nicht nutzbar oder nur mit einem gewissen Risiko. Aktuell wird noch daran gearbeitet die Bugs, die bei der Nutzung von ioBroker mit Node.js 7.x und höher auftreten, zu beheben. Deshalb wird auf die Nutzung des EnOcean-Adapters in dem exemplarischen Smart Home Szenario dieser Thesis verzichtet.

Nach der Installation des **Z-Wave**-Adapters über das Web User Interface, kann für diesen unter dem Reiter Adapter die Konfiguration durchgeführt werden. In dem Konfigurationsfenster muss zunächst der USB-Pfad zu dem Z-Wave Stick angegeben werden (`/dev/ttyACM0`). Anschließend kann unter dem Reiter Geräte der „Geräte hinzufügen “-Button gedrückt werden. Damit wird ioBroker in den Pairing-Mode versetzt und es kann der Pairing-Knopf des Z-Wave Sensors gedrückt werden. Der gefundene Sensor wird daraufhin unter Objekte als Datenpunkt unter dem Z-Wave Objekt angezeigt und kann genutzt werden.

Auch für die **MAX! Cube Heizungssteuerung** ist ein Adapter in ioBroker vorhanden. Nach der Installation müssen in der Konfiguration lediglich IP-Adresse und Port (62910) angegeben werden und das Geräte kann verwendet werden.

Für die **Yeelight LED** stellt ioBroker noch keinen Adapter zur Verfügung. In den ioBroker Foren wurde dieses Problem schon behandelt und ein Shell-Script von einem User erzeugt, mit dem über einen kurzen Umweg die LED von ioBroker aus gesteuert werden kann. Auf die Einrichtung der Yeelight wird, da für sie von ioBroker aus kein Adapter zur Verfügung gestellt wird, im Unterkapitel

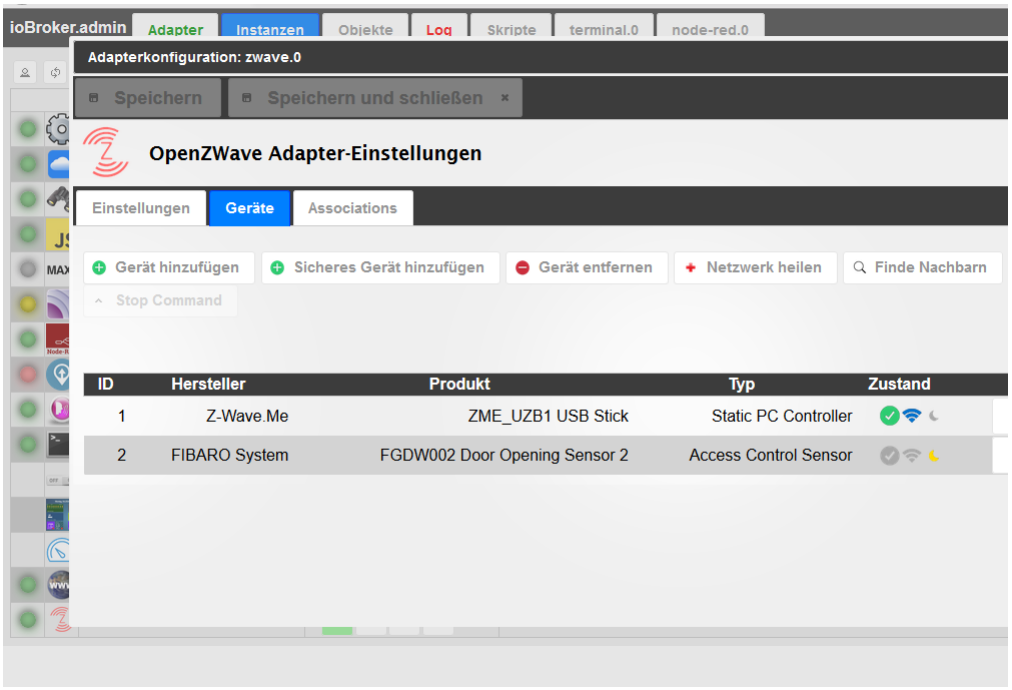


Abbildung 4.21: ioBroker Z-Wave Pairing

Erweiterbarkeit eingegangen.

Über den „**MQTT** Broker/Client“-Adapter kann mit geringem Aufwand ein MQTT Server in ioBroker eingerichtet werden. Nach der Installation des Adapters wird der MQTT Adapter unter Instanzen konfiguriert. Um die Daten des, als MQTT Client fungierenden, ESP32 zu erhalten, wird in ioBroker mit dem MQTT Adapter ein Broker eingerichtet. Mit Angabe des Ports, des User-Namen und des Passworts kann der MQTT Broker konfiguriert werden. Für die verschiedenen Topics werden automatisch in der Objekte-Ordnerstruktur Datenpunkte angelegt und die Daten können nun in der Visualisierung ausgegeben oder für die Automation weiterverwendet werden.

Um ioBroker mit dem **Sprachassistenten** Amazon Alexa zu verknüpfen, muss der Cloud-Adapter installiert und konfiguriert werden. Hierfür wird ein APP-Key benötigt. Dazu muss zunächst unter https://iobroker.net ein Benutzerkonto angelegt werden. Dort kann, per Klick auf einen Button, der Key erstellt werden. Dieser ist eine Kombination aus der Email-Adresse des Benutzerkontos und mehreren Buchstaben und Zahlen.

Der APP-Key muss im Konfigurationsfenster des Cloud-Adapter angegeben werden. Damit ist eine Verbindung zwischen dem lokalen ioBroker und dem Internet, genauer gesagt dem Server, welcher beispielsweise für Amazon Alexa

benötigt wird, hergestellt. In der Amazon-Alexa-App muss der ioBroker-Skill installiert und mit dem zuvor angelegten ioBroker-Benutzerkonto über die Login-Daten verknüpft werden.

Abbildung 4.22: ioBroker Cloud-Konfiguration

Befehle an Amazon Alexa werden nun an die Server von Amazon geschickt. Dort werden sie verarbeitet und mit dem, über die Alexa-App installierten, ioBroker Skill an den ioBroker Server geschickt. Dieser schickt den Befehl über den Cloud-Adapter an den lokalen ioBroker zu Hause, welcher letztendlich den Befehl ausführt.

Das Einrichten der Smart Home Geräte für die Nutzung mit Alexa findet in der Konfiguration des Cloud-Adapters statt. In dieser können, unter dem Reiter *Smarte Geräte*, Datenpunkte aus der Objekte-Ordnerstruktur ausgewählt werden (beispielsweise der Status eines Objektes). Danach wird dieser Datenpunkt in der Amazon-Alexa App bei der Suche nach neuen Geräten gefunden und kann fortan mit seinem „Smartname“ angesteuert werden.

Für die Verwendung von **OwnTracks** stellt ioBroker einen weiteren Adapter bereit, bei dem noch IP-Adresse, Port, Anwender und Passwort angegeben werden müssen. Anschließend sind die GPS-Daten, die von der Smartphone-App über MQTT versendet werden, in ioBroker verwendbar.

Eine allgemeine **ZigBee**-Erweiterung stellt ioBroker nicht zur Verfügung. Dafür einige herstellerspezifische Adapter, welche über ZigBee kommunizieren, wie etwa ein Philips Hue Bridge-Adapter.

Ein **HTTP**-Adapter ist bei ioBroker auch nicht verfügbar. Allerdings kann über den Umweg der Node-RED Erweiterung, die auch für das Erstellen von Automationen in ioBroker verwendet wird, eine HTTP-Node installiert und

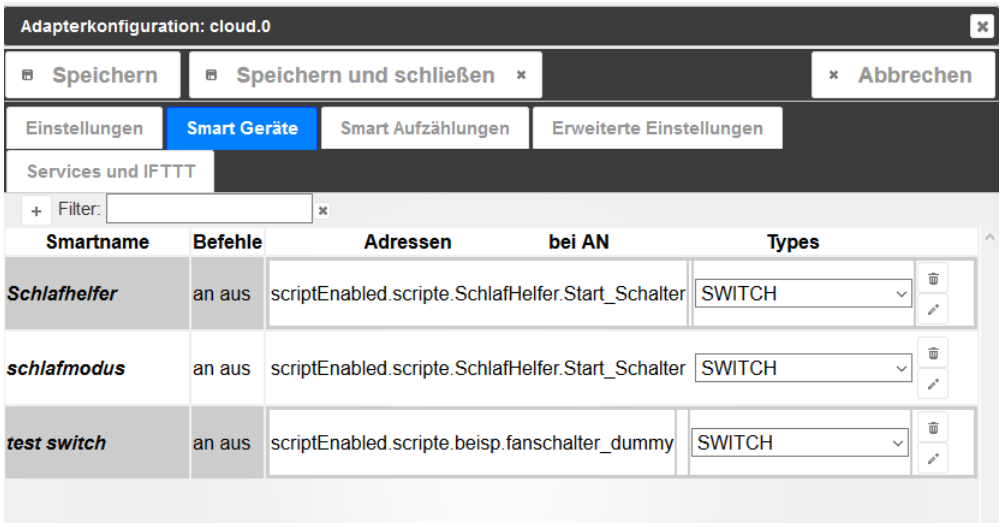


Abbildung 4.23: ioBroker Smarte Geräte

über diese beispielsweise ein HTTP-Request versendet werden.

Für die Kommunikation mit **Bluetooth** Low Energy Geräten stellt ioBroker einen Adapter bereit.

Visualisierung

Über den Habpanel-Adapter lässt sich schnell eine blockbasierte Bedienoberfläche in ioBroker erstellen. Besonders stolz sind die Macher von ioBroker allerdings auf den Vis-Adapter, über den sich eine professionelle Bedienoberfläche für das Smart Home erstellen lässt. Die Nutzung des Vis-Adapters ist für den privaten Gebrauch kostenlos, benötigt allerdings einen Lizenzschlüssel, welcher nach der Registrierung unter `iobroker.net` erstellt werden kann. Dieser muss bei der Konfiguration des Vis-Adapters angegeben werden. Über die Instanz des Vis-Adapters in der Weboberfläche von ioBroker, gelangt man zu dem Web-Editor (`http://raspberrypi-IP:8082/vis/edit.html`) in dem die Visualisierung umgesetzt werden kann.

Die Widgets werden per Drag & Drop in die Seite gezogen und dort konfiguriert bzw. ausgewählt, welches Objekt über diese gesteuert werden soll. Im Web-Editor kann eine Vielzahl von Zusatzpaketen mit Icons und Widgets heruntergeladen und im Web-Editor genutzt werden. Zusätzlich besteht die Möglichkeit, eigene Grafiken in den Web-Editor einzubinden. So kann ein Bild des Grundrisses der Wohnung genutzt werden und in diesem Icons für die Geräte (Beispielsweise Glühbirnen-Icons für Lampen) positioniert und angesteuert

werden. Ebenso könnte der Status von intelligenten Sensoren visualisiert werden, etwa durch ein geöffnetes oder geschlossenes Fenster im Gebäudegrundriss. Über CSS können die Grafiken und Widgets noch genauer definiert und bearbeitetet werden (bspw. Position, Farben, Staffelung mittels Z-Index). Durch die logische Struktur und das Konfigurieren der einzelnen Geräte direkt in der Weboberfläche, lässt sich so auf einfache Art und Weise eine ansprechende Steuerzentrale für das Smart Home entwerfen. Eine einfache Bedienung über Kacheln für Tablet und Smartphones lässt sich innerhalb von kurzer Zeit erstellen.

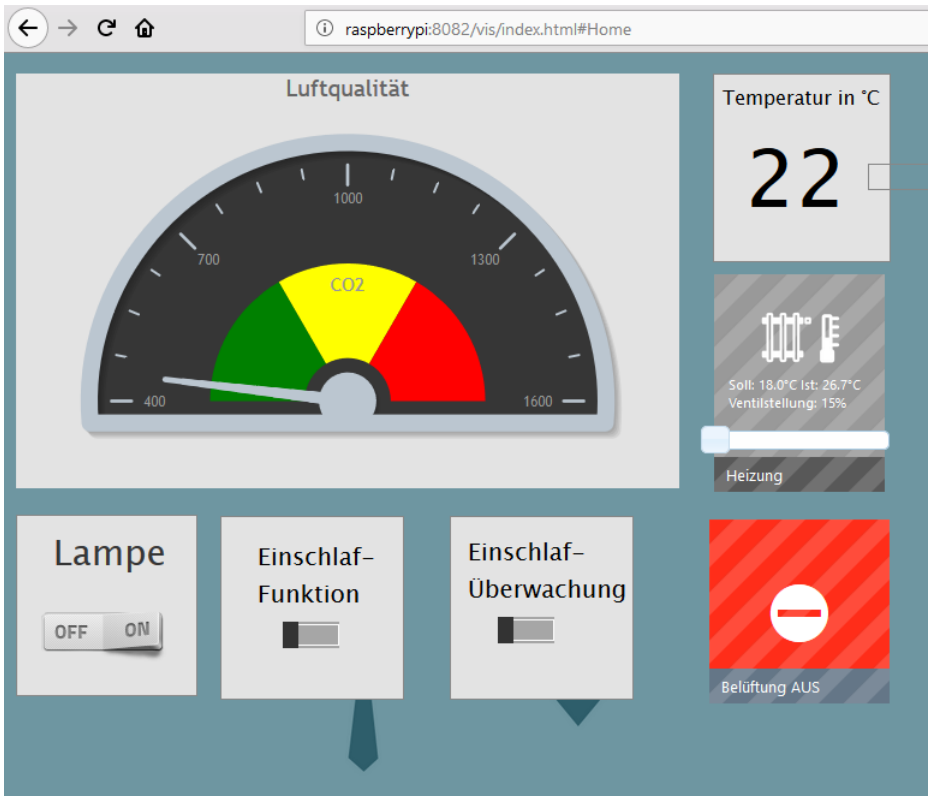


Abbildung 4.24: ioBroker einfache Bedienoberfläche

Ein Kritikpunkt stellt die Tatsache dar, dass die Auflösung der Bedienoberfläche zuvor festgelegt werden muss und so ein responsives Verhalten, also die stufenlose, passende Anzeige bei verschiedenen Displaygrößen, nicht möglich ist.

Abbildung 4.25: ioBroker Vis Grundriss Demo

Erweiterbarkeit

Für das Ausführen von externen Scripten, aus der Bedienoberfläche von ioBroker heraus, muss sich eines Tricks bedient werden. Zunächst muss der JavaScript-Adapter installiert werden. Dieser wird auch für die spätere Erstellung der Automation genutzt. Eine leere Script-Datei wird erstellt. Ein Script hat die Zustände „an“ und „aus“. Diese Tatsache wird genutzt, um das leere Script als Dummy-Schalter zu verwenden. Für die Ausführung des externen Scripts wird eine weitere Script-Datei angelegt. Diese wird mit der visuellen Programmierumgebung Blockly, welche mit dem JavaScript-Adapter mitinstalliert wird, bearbeitet. Blockly ermöglicht es, Code durch das Zusammenfügen von visuellen Blöcken zu erstellen. Dieser „visuelle“ Code wird von Blockly anschließend

in JavaScript-Code umgewandelt.

In dem Blockly-Script wird ein Trigger-Block gewählt, der über die Objekt-ID den Zustand des Dummy-Schalters überwacht. Wechselt der Zustand des Schalters von „aus“ zu „an“ wird der in Blockly integrierte Exec-Block ausgeführt. Mit diesem kann über die Angabe des Pfades der zuvor installierte Service *raspberry-remote* gestartet und mit diesem die Funksteckdose über das 433 MHz Sendemodul angesprochen werden.

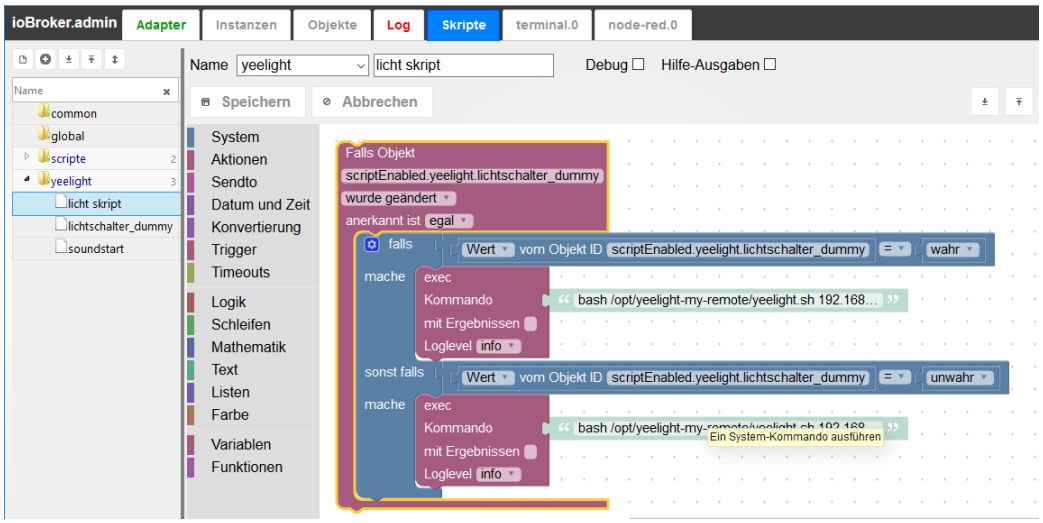


Abbildung 4.26: Verwendung Exec-Kommando für Yeelight

Auf die gleiche Weise wird ein Shell-Script, das die Steuerung der Yeelight LED übernimmt, angesprochen.

Für das Einbinden von selbst geschriebenen Adaptern oder die Nutzung von Adaptern, welche von anderen Nutzern auf GitHub bereitgestellt wurden, bietet ioBroker in seiner Web-Oberfläche einen Button. Über diesen können Adapter aus eigener URL, beispielsweise lokal aber auch von GitHub oder anderen Filehosting-Plattformen, installiert werden. Eigene Adapter für ioBroker werden mit JavaScript erstellt. Eine ausführliche Anleitung auf GitHub gibt dem Entwickler eine Hilfestellung bei der Entwicklung von eigenen Adaptern.

Automation

Für die Erstellung von Automationen bietet ioBroker verschiedene Möglichkeiten. Drei der meist genutzten sind das Programmieren der Abläufe mittels JavaScript und die visuelle Programmierung über Blockly oder Node-RED.

Visuelle Programmierung soll Programmier-Neulingen den Einstieg in das Programmieren erleichtern. Um die Möglichkeiten von der ioBroker-Automation auszutesten, werden für die Umsetzung der Anforderungen des Smart Home Szenarios alle drei Möglichkeiten eingesetzt.

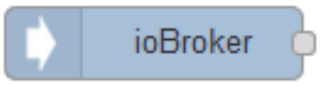
Für die Erstellung von Automationen müssen zunächst die entsprechenden Adapter (JavaScript-Adapter für die Nutzung von JavaScript und Blockly und Node-RED-Adapter für die Erstellung von flow-basierten Automationen) installiert werden.

Nach der Installation können die Adapter als weitere Reiter-Elemente zu der ioBroker Weboberfläche hinzugefügt werden. Unter dem Reiter *Script* kann in einer Ordnerstruktur der Ablage-Ort des Scriptes festgelegt und ein neues Script-File angelegt werden. Für die Automatisierung mittels Node-RED wird ein eigener Reiter in der Weboberfläche angezeigt. Hier können verschiedene Flows für die einzelnen Automationen angelegt werden.

Umsetzung Szenario

Anforderung I:

Für die Umsetzung von Anforderung I wird der Node-RED Adapter von ioBroker verwendet, um die Automation visuell mit flowbasierten Blöcken zu erstellen. Über eine ioBroker-Node, die sich links in der Palette der Node-RED-Oberfläche befindet, werden die in ioBroker erstellten Objekte eingebunden.



In der Node-Konfiguration können aus einer Ordnerstruktur die entsprechenden Datenpunkte, die angesteuert, überwacht oder ausgelesen werden sollen, ausgewählt werden. Für die Umsetzung der ersten Anforderung, dem Starten der Einschlaffunktion mittels des Sprachassistenten Alexa, wird ein Schalter, der in ioBroker mit Alexa gesteuert wird, als Datenpunkt in der ioBroker-Node gewählt.

Anforderung II:

Das Starten des Einschlafassistenten mittels EnOcean-Button ist aufgrund der noch fehlenden Kompatibilität von ioBroker mit Node.js 7.x nicht möglich.

Anforderung IV

Die mit Alexa verknüpfte ioBroker-Node wird mit einer „function“-Node verbunden. Diese ermöglicht das Schreiben von kleinen JavaScript-Codeblöcken. In der „function“-Node wird überprüft, ob eine Variable, die den Dimm-Wert festlegt, größer als Null ist. Ist dies der Fall, wird dieser Wert an eine verbun-

dene Exec-Node weitergeleitet, welche die smarte Yeelight LED ansteuert und die Dimmer-Variable heruntergezählt.

```
if(context.global.dimmVal >= 0){
    msg.payload = context.global.dimmVal;
    context.global.dimmVal
        = context.global.dimmVal - 1;
return msg;
}
```

Zusätzlich wird die „function“-Node nach einer gewissen Verzögerung, die mit einer „delay“-Node ermöglicht wird, erneut aufgerufen. In dieser wird erneut der Wert der Variablen untersucht, welche die Helligkeit festlegt. Ist dieser weiterhin größer bzw. gleich Null, wird der nächste Dimm-Schritt eingeleitet.

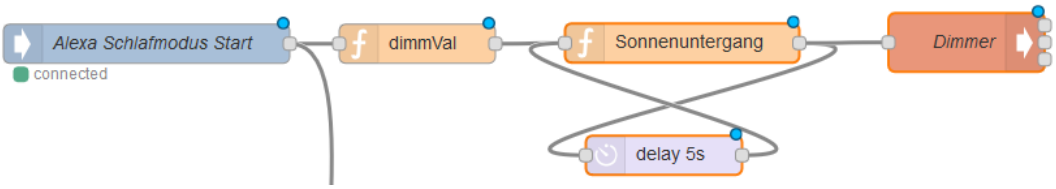


Abbildung 4.27: ioBroker Dimmer

Anforderung V:

Eine delay-Node wird zusätzlich zum Starten der Schlafüberwachung genutzt und 30 Minuten nach der Aktivierung der Einschlaffunktion durch Alexa automatisch die Schlafüberwachung gestartet.

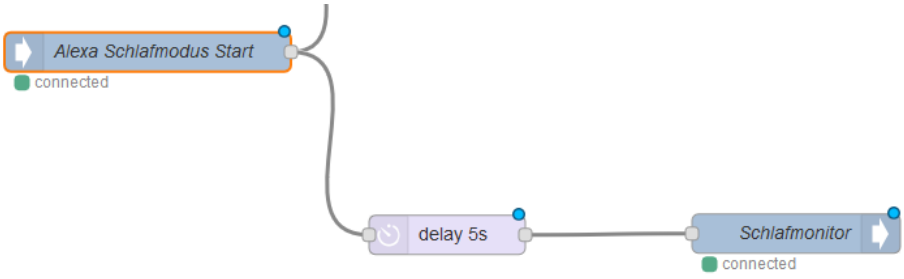


Abbildung 4.28: ioBroker Schlafüberwachung starten

Anforderung III:

Das Abspielen der Sound-Datei im Schlafmodus wird mittels Blockly implementiert. Hier wird auch der Objekt-Status des Alexa-Schalters überwacht. Ändert

sich dieser, wird überprüft, ob der Schalter nun den Wert „wahr“ hat. Ist dies der Fall, wird der installierte Sayit-Adpater gestartet, über welchen Sound ausgegeben werden kann.

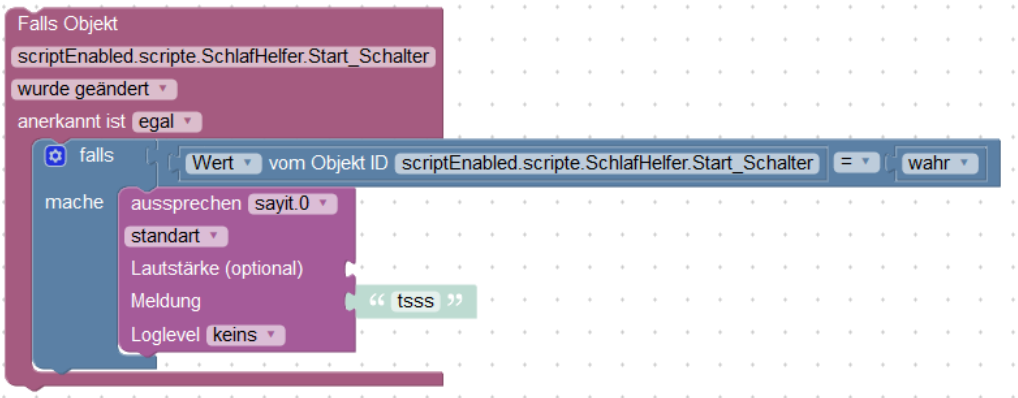


Abbildung 4.29: ioBroker Sound-Datei abspielen

Anforderung VI:

Die Heizungssteuerung im Schlafmodus wird ebenfalls mit Blockly erstellt. Hier wird bei einem Start des Schlafmodus das Thermostat auf 18 Grad gesetzt.

Anfoderung VII -VIII:

Die Überwachung des CO2-Wertes wurde direkt mit JavaScript implementiert. Es wird auf Veränderungen der übermittelten CO2-Werte geschaut.

```
on({id: "mqtt.0.room.airquality.eco2"
    ,change: "ne"}, function (obj) {
```

Ändern sich diese wird geschaut, ob der Schlafmodus aktiviert ist und überprüft, ob der CO2-Wert den Grenzwert von 600 überstiegen hat.

```
if(getState("javascript.0.scriptEnabled.script
    .SchlafHelfer.Schalter_Schlafmonitor").val
    && (getState("mqtt.airquality.eco2").val)>600)){
```

Trifft beides zu, wird der *raspberry-remote*-Service genutzt, um den Ventilator, also die Belüftung, zu starten.

```
exec(' /opt/raspberry-remote/send 11111 1 1');
```

Die Temperatur wird auf dieselbe Art und Weise per JavaScript überwacht. Der Status des Fenstersensors kann mit der Funktion `getState()` abgefragt werden.

```
if(getState("zwave.0.NODE2.ALARM.Access_1").val == 23)
//value 23 -> window closed
//value 22 -> window opened
```

Anfoderung IX - XIII:

Die Aufweckfunktion wird komplett mit dem Node-RED-Adapter umgesetzt. Zum Start der Aufweckfunktion wird eine Inject-Node verwendet. Mit dieser kann zu einer festgelegten Uhrzeit eine Nachricht an die nächste Node versendet werden. Hier wird zunächst der Schlafmodus durch Änderung des Status des Schlafmodus-Schalters deaktiviert. Der Sonnenaufgang wird nach dem gleichen Prinzip wie der Sonnenuntergang umgesetzt.

Für das Setzen der Temperatur wird eine „function“-Node genutzt. Diese passt das msg-Objekt, welches die Informationen zwischen den einzelnen Nodes transportiert, entsprechend an, indem es den Payload auf 24 setzt. Dieser Wert wird an die ioBroker-Node weitergeleitet, welche mit dem Temperatur-Datenpunkt des Heizkörper-Thermostats verbunden ist.

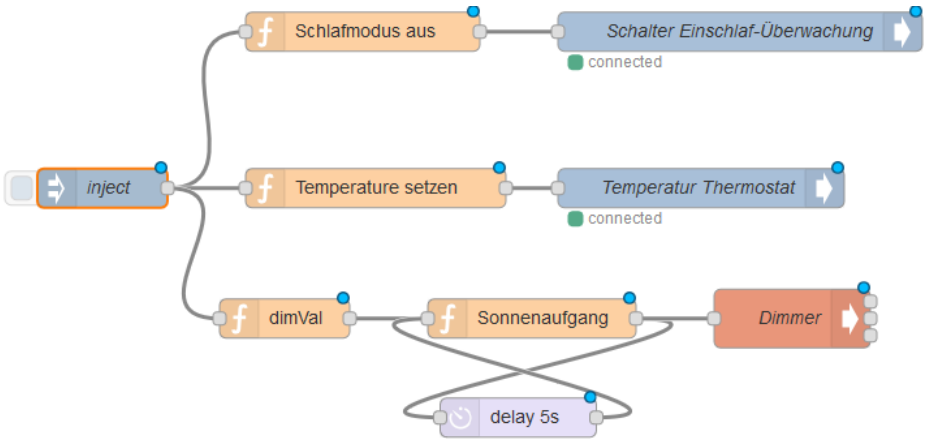


Abbildung 4.30: ioBroker Aufweckfunktion

Anforderung XIV:

Aufgrund zeitlicher Engpässe konnte die letzte Anforderung, das Aussetzen der Aufweckfunktion, wenn der Nutzer nicht zu Hause ist, nicht mit in die Automation aufgenommen werden.

Das Einbinden der Lokalisierung mittels OwnTracks-App in die Smart Home Plattform funktionierte aber bereits problemlos. Dies legt die Vermutung nahe, dass auch diese Anforderung bei mehr Zeitreserven hätte umgesetzt werden können.

4.5.3 Home Assistant

Installation

Home Assistant bietet verschiedene Möglichkeiten zur Installation und Nutzung der Software-Plattform auf dem Raspberry Pi. Der klassische Weg ist die manuelle Installation. Hierzu muss auf dem Raspberry Pi zunächst die Raspbian Lite Distribution installiert werden. Diese Art der Installation ist die komplexeste, da die Installation aus einer virtuellen Python-Umgebung heraus erfolgen muss. Dafür müssen zunächst einige Python Packages heruntergeladen, ein Home Assistant Account und eine neue Directory erstellt und Rechte zugewiesen und anschließend die virtuelle Umgebung eingerichtet werden. Ebenso muss auch das automatische Ausführen von Home Assistant beim Start des Raspberry Pi bei dieser Installations-Variante noch selbst eingerichtet werden.

Alternativ bietet Home Assistant mit Hass.io und Hassbian zwei vorgefertigte Raspbian-Distributionen. Diese werden auf die SD-Karte des Raspberry Pi geflasht und bieten eine angepasste Raspbian-Version, in der Home Assistant bereits installiert und vorkonfiguriert ist. Hass.io ist die neueste Distribution, die von Home Assistant bereitgestellt wird. Diese wird als besonders anfängerfreundlich beschrieben und hat das Ziel, die Nutzung von Home Assistant unter komplettem Verzicht auf die Kommando-Zeile zu ermöglichen. Konfigurationen und Einstellungen werden vollständig in der Web-Oberfläche von Home Assistant durchgeführt. Diese Version bietet eine sehr unkomplizierte Nutzung, schränkt den Nutzer allerdings auch ein. So können in dieser Distribution beispielsweise keine weiteren Services und Programme, außerhalb von Home Assistant, direkt auf dem Raspberry Pi installiert und genutzt werden, da man keinerlei Admin-Rechte besitzt, um weitere Pakete zu installieren.

Einen Kompromiss stellt Hassbian dar. Hier handelt es sich ebenfalls um eine modifizierte Raspbian Lite Version, in der Home Assistant installiert und vorkonfiguriert ist. Allerdings hat man weiterhin auch Zugriff auf die Linux-Distribution, kann diese über die Kommando-Zeile weiter konfigurieren und auch Debian-Software-Pakete auf dem Raspberry Pi nutzen und auf diese aus der Home Assistant Bedienoberfläche hinaus zugreifen.

Oberfläche

Die Web-UI von Home Assistant wird geöffnet über `http://ipRaspberry:8123`. Das Web-Frontend wurde mit der JavaScript Library Polymer 2.0 erstellt. Auf

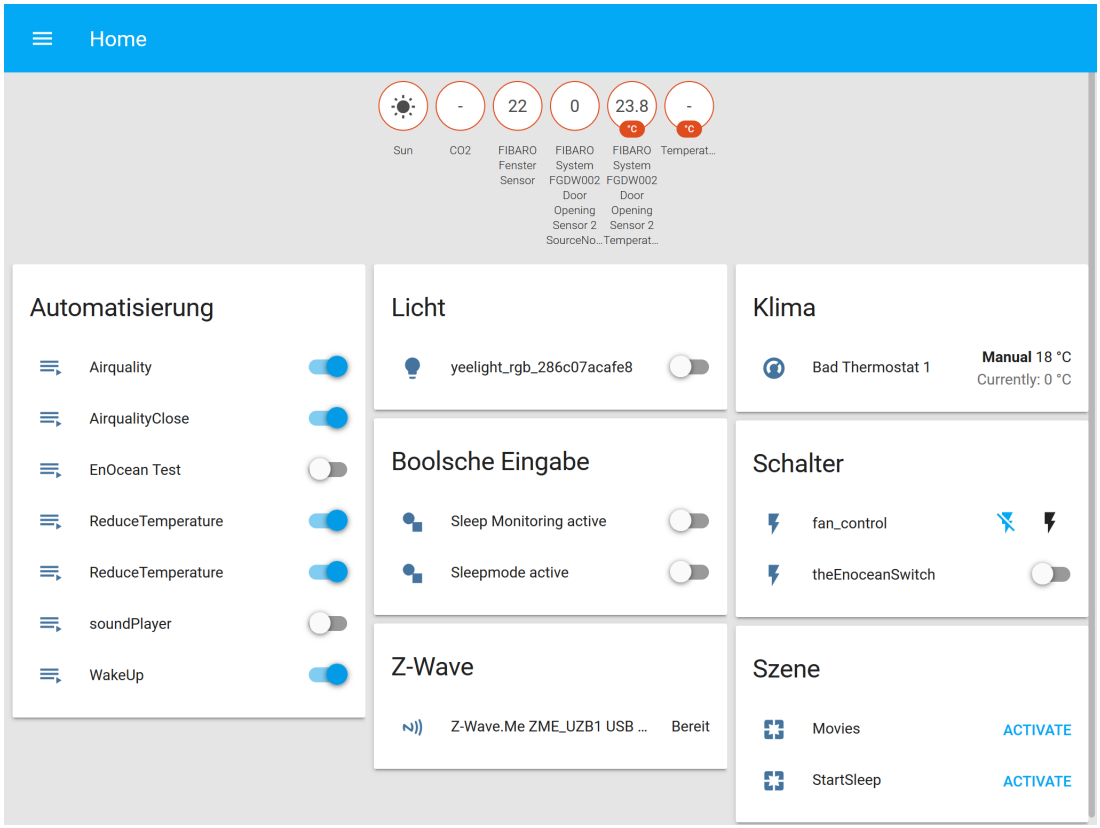


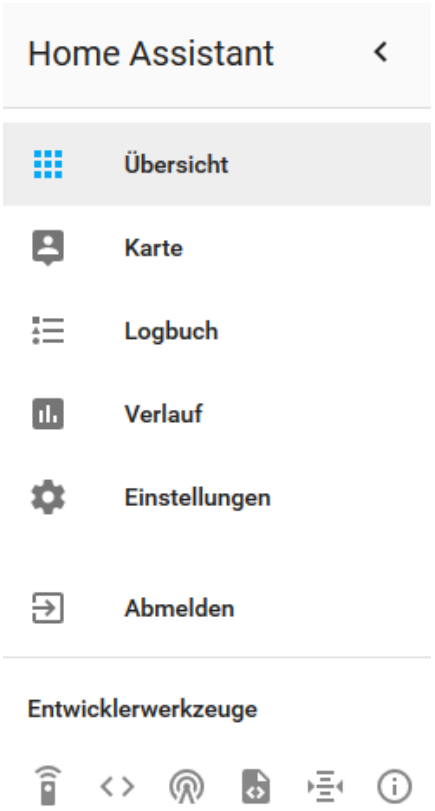
Abbildung 4.31: Home Assistant Web-UI

der Startseite werden die installierten Geräte und Dienste angezeigt. Diese dient auch als Bedienoberfläche für das Smart Home. Über das Burger-Icon links oben klappt sich eine Navigation auf, über welche man zu weiteren Fenstern für die Konfiguration und Steuerung von Home Assistant gelangt.

Über den Link **Logbuch** werden Aktivitäten protokolliert, wie beispielsweise das Einschalten eines der Geräte. Unter **Verlauf** werden zeitliche Statusveränderungen der verschiedenen Geräte veranschaulicht. Über den Reiter **Einstellungen** kann sich z.B. in das Home Assistant Nutzer Konto eingeloggt oder die Konfiguration überprüft werden [88].

In den Entwicklerwerkzeugen besteht die Möglichkeit, Dienste (Fernbedienungs-Icon), wie ein Neustart des Home Assistant Servers zu nutzen. Unter Zustände (<>) werden der aktuelle Status und Attribute der eingebundenen Geräte und Dienste aufgelistet [88].

Für das Hinzufügen und Konfigurieren von neuen Geräten und Diensten muss auf einen Text-Editor zurückgegriffen werden und die Konfigurationsdatei `configuration.yaml` angepasst werden. Mehr dazu im nachfolgenden Kapitel Konfiguration.



Konfiguration

Die Konfiguration und das Hinzufügen von Smart Home Geräten und Diensten findet in einer bestimmten Konfigurationsdatei namens `configuration.yaml` statt. Diese befindet sich unter `~/.homeassistant`. Zu beachten ist, dass nur der User „homeassistant“ Zugriff auf die Datei hat und nicht der Nutzer „pi“. Mit dem Befehl

```
1 $ sudo su -s /bin/bash homeassistant
```

kann der Nutzer gegebenenfalls gewechselt werden.

Für die Bearbeitung der Konfigurationsdateien bietet es sich an einen Samba-Server einzurichten und die Ordner des Nutzers homeassistant im Netzwerk freizugeben. So können die Dateien über einen PC in einem Editor geöffnet werden, um diese zu bearbeiten. Dies ist deshalb besonders wichtig, da die verwendete Markup-Language YAML auch Einrückungen im Code beachtet.

Bei dem ersten Start von Home Assistant wird die `configuration.yaml` angelegt. In dieser werden das Web Interface und die Discovery-Funktion, also

das automatische Finden neuer Geräte, aktiviert. So beginnt Home Assistant beim ersten Start schon einen Suchlauf nach Smart Home Geräten im Netzwerk und fügt die gefundenen Geräte zur Weboberfläche hinzu [89]. Beim ersten Start wurde so die smarte Yeelight LED automatisch gefunden, eingerichtet und konnte sofort in der Web-UI verwendet werden.

Bei Änderungen der `configuration.yaml` muss Home Assistant neu gestartet werden, damit diese in Kraft treten. Über den Kommandozeilen-Befehl `hass –script check config` können die Änderungen vor dem Neustart getestet werden [89].

Für das Einbinden von weiteren Components in Home Assistant muss der entsprechende Code zu der `configuration.yaml` hinzugefügt werden. Die Syntax der YAML-Datei besteht aus einzelnen Blöcken für die Geräte oder Geräte-Gruppen und Key-Value-Paaren [90].

Die Code-Schnipsel für das Einbinden neuer Components sind online zu finden unter `https://home-assistant.io/components/`. Unter diesem Link findet man über 900 verschiedene Dienste und Geräte. Zu jedem verfügbaren Component wird der Code-Schnipsel, der hinzugefügt werden muss, sowie eine Anleitung für die Konfiguration, bereitgestellt.

Für die Konfiguration der Components muss das Code-Snippet nach dem Kopieren in die `configuration.yaml` entsprechend angepasst, genauer gesagt die Value Werte für die benötigten Keys, definiert werden.

Für die Nutzung des **EnOcean** Components muss der Pfad zu dem USB-Gerät angegeben werden (`device: /dev/ttyUSB0`). Für das Einrichten des Schaltmoduls wird ein „binary_sensor“-Component angelegt. Durch Angabe der Plattform (*enocean*) weiß Home Assistant, um welche Art von Gerät es sich handelt. Zusätzlich muss die ID des Schaltmoduls, welche auf dem Gerät selbst aufgedruckt ist, mitangegeben werden, damit Home Assistant die empfangenen Daten dem Gerät zuordnen kann.

```
enocean:
    device: /dev/ttyUSB0

binary_sensor:
    -   platform: enocean
        id: [0xXX,0xXX,0xXX,0xXX]
        name: living_room_switch
```

Um **Z-Wave**-Geräte zu nutzen, muss folgender Eintrag zu der `configuration.yaml` hinzugefügt werden:

```
zwave:
    usb_path: /dev/ttyACM0
```

Mit `/dev/ttyACM0` wird der Pfad zu dem Z-Wave USB-Stick angegeben.

Bei dem Aktivieren der Pairing-Funktion des Z-Wave Sensors wird der Sensor umgehend von Home Assistant entdeckt, das Gerät hinzugefügt und entsprechende Elemente für die Visualisierung der Daten der Web-Oberfläche hinzugefügt.

Die **Yeelight** LED benötigt keine weitere Konfiguration, da diese beim ersten Start von Home Assistant automatisch entdeckt wird und sofort betriebsbereit ist.

Für die Nutzung des **MAX! Cube Heizkörperthermostats** muss lediglich die IP-Adresse des Cubes angegeben werden. Die Kopplung mit dem Thermostat findet automatisch statt und die entsprechenden Datenpunkte (Setzen der Temperatur, Aktuelle Temperatur des Thermostats) werden in der Bedienoberfläche angezeigt.

```
maxcube:
    host: 192.168.x.x
```

Das Einbinden von **MQTT** gestaltet sich ähnlich einfach. Durch die Angabe von Broker-IP-Adresse, dem Port, Nutzer und Passwort ist die Component fertig eingerichtet. Um die Topics zu abonnieren und in der Web-UI anzeigen zu lassen, müssen Sensoren angelegt werden. Durch die Angabe, welche Plattform genutzt wird (mqtt), eines selbst gewählten Namens und welches Topic abonniert werden soll, ist die grundsätzliche Konfiguration abgeschlossen. Es besteht die Möglichkeit, optionale Angaben wie Maßeinheiten (hier Grad Celsius) hinzuzufügen.

```
mqtt:
    broker: 192.168.x.x
    port: 1883
    client_id: home-assistant-1
    username: user1
    password: password

sensor:
    -   platform: mqtt
        state_topic: 'room/airquality/temperature'
        name: 'Temperature'
        unit_of_measurement: 'C'
```

```
    - platform: mqtt
      state_topic: 'room/airquality/eco2'
      name: 'CO2'
```

Für die Nutzung des Alexa **Sprachassistenten** in Home Assistant muss in der `configuration.yaml` folgender Eintrag vorgenommen werden:

```
cloud:
```

Nach einem Neustart findet man in der Weboberfläche unter *Configuration* nun einen Button mit der Aufschrift „Home Assistant Cloud“. Über diesen kann ein Home-Assistant-Benutzerkonto erstellt werden. In der Amazon-Alexa App muss der Home Assistant Skill heruntergeladen werden. Dieser ist in Deutschland noch nicht verfügbar. Allerdings besteht die Möglichkeit, die Echo-Region seines Amazon-Kontos auf USA zu stellen. Diese Einstellung kann auf Amazon.de unter *Mein Konto* > *Digitale Inhalte* > *Meine Inhalte und Geräte* angepasst werden. Anschließend ist der Skill in der Alexa-App verfügbar. In dem Home Assistant-Skill muss sich wiederum mit den Home Assistant Benutzerkonto-Daten eingeloggt werden. Anschließend kann in der Alexa-App nach neuen Geräten gesucht werden. Alle mit Home Assistant verknüpften Geräte werden nun angezeigt. Soll gefiltert werden, welche Geräte von der Alexa-App gefunden werden, muss zu dem entsprechenden Component in der `configuration.yaml` noch folgendes Key-Value-Paar hinzugefügt werden:

```
alexa_hidden: true
```

So gekennzeichnete Elemente werden von der Alexa-App nicht entdeckt. Um den Sprachassistenten im Smart Home Szenario zu nutzen, wird zusätzlich eine Art Dummy-Button angelegt, dessen Status über Alexa verändert wird. Dazu wird ein Component vom Typ „input_Boolean“ in der `configuration.yaml` angelegt.

```
input_boolean:
    sleep_mode:
        name: Sleepmode active
        initial: off
```

Dieser wird von der Alexa-App gefunden und über den Sprachassistenten kann der Zustand des Dummy-Buttons angepasst und dieser als Trigger für eine Automation verwenden werden.

Folgender Eintrag ermöglicht die Nutzung von OwnTracks in Home Assistant. Anzupassen ist hier das MQTT-Topic, entsprechend dem in der Smartphone-App festgelegten Wert.

```
device_tracker:
    -   platform: owntracks
        max_gps_accuracy: 200
        waypoints: True
        mqtt_topic: "owntracks/#"
```

Visualisierung

Die Bedienoberfläche von Home Assistant ist gleichzeitig auch die Startseite der Web-UI. Sie besteht aus mehreren Kacheln, die nach dem Einbinden eines Component automatisch zu der Oberfläche hinzugefügt werden. Soll ein Component nicht in der Bedienoberfläche erscheinen, muss folgender Befehl in der `configuration.yaml` bei dem jeweiligen Gerät oder Dienst hinzugefügt werden:

```
hidden: true
```

Die einzelnen Kacheln der Bedienoberfläche werden State Cards genannt. Sie zeigen den Namen der Entity, ein bestimmtes Icon und den aktuellen Status des Components. Sensoren werden nicht als State-Card, sondern als kleine Kreise mit Bild, Namen und Status, als so genannte Badges, oberhalb der State Cards angezeigt. Eine Individualisierung der Bedienoberfläche ist durch das Einbinden eigener State Cards als HTML-Dateien möglich [91]. Zusätzlich bietet Home Assistant die Möglichkeit, mit der Installation von „HA-Dashboard“ eine weitere Bedienoberfläche einzurichten. Die Installation des Dashboards ist allerdings mit etwas Aufwand verknüpft. Eine detaillierte Anleitung für die Installation wird von Home Assistant bereit gestellt. Das Dashboard in Kachel-Optik ist speziell für die Steuerung mittels „wall mounted“-Devices gedacht, wie etwa einem Tablet, das als Steuereinheit an der Wand montiert ist [18].

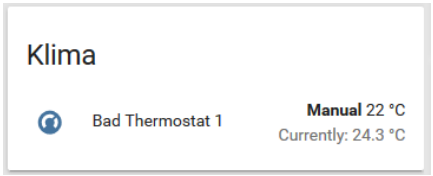


Abbildung 4.32: HA State Card

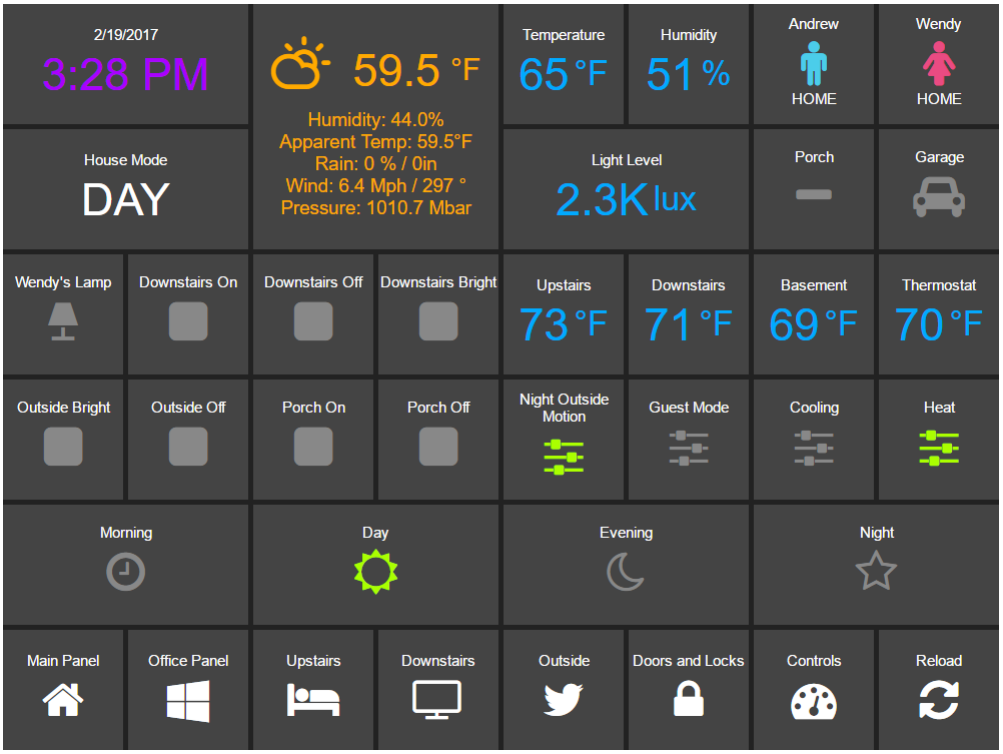


Abbildung 4.33: Home Assistant Dashboard [18]

Erweiterbarkeit

Das Ausführen von externen Scripten zeichnet sich durch seine einfache Konfiguration aus. Hierzu muss lediglich ein Schalter (Switch-Component) angelegt und zu dem „platform“-Key der Wert `command_line` hinzugefügt werden. Bei den Angaben `command_on` und `command_off` kann der jeweilige Befehl angegeben werden.

```
switch:
  - platform: command_line
    switches:
    fan_control:
    command_on: "/opt/raspberry-remote/send 11111 1 1"
    command_off: "/opt/raspberry-remote/send 11111 1 0"
```

Components werden in Home Assistant in der Programmiersprache Python geschrieben. Für das Erstellen eigener Components stellt Home Assistant diverse Tutorials bereit. Für das Einbinden weiterer Components, etwa von GitHub,

müssen diese auf dem Raspberry unter `~/.homeassistant/custom_components` abgelegt werden. Bei einem Start von Home Assistant durchsucht dieser den Pfad nach neuen Components [92]. In der `configuration.yaml` können diese anschließend konfiguriert werden.

Automation

Für Automationen stellt Home Assistant einen weiteren YAML-File bereit (`automations.yaml`). Diese Textdatei befindet sich unter `~/.homeassistant`. In Home Assistant besteht eine Automation, wie schon in 3.2.3 beschrieben, aus drei Teilen: Einem Trigger, der die Automation startet, einer oder mehrere Conditions, also Bedingungen, die überprüft werden, bevor der dritte Teil, die Action ausgeführt wird. Eine Automation könnte dann folgendermaßen aussehen:

```
alias: 'ReduceTemperature'
trigger:
    platform: numeric_state
    entity_id: sensor.temperature
    below: 21
condition:
-   condition: state
    entity_id: input_boolean.notify_home
    state: 'on'
action:
- service: switch.turn_off
    data:
        entity_id: switch.fan_control
```

Die Angabe hinter „platform“ bestimmt die Art des Triggers. Dieser kann beispielsweise ein State-Trigger, ein Numeric-State-Trigger (überwacht Sensor-Daten) oder ein Time-Trigger sein, der zu einer bestimmten Uhrzeit ausgelöst wird [93].

Zusätzlich bietet Home Assitant die Option, über einen Automation-Editor intelligente Abläufe direkt in der Web-Oberfläche zu definieren. Dazu muss in der Web-UI der Menüpunkt *Configuration* gewählt werden. In diesem kann *Automation* ausgewählt werden, um den Automation-Editor zu starten. Hier wird nun schrittweise durch die Automations-Erstellung geleitet [94].

Umsetzung Szenario

Anforderung I - VI:

Die Anfoderungen des Einschlafassistenten können in einer Regel definiert werden. Es können mehrere Trigger für die Regel festgelegt werden. Diese sind für den Einschlafassistenten die Statusänderung des Input-Booleans, der über Alexa gesteuert wird oder das Drücken eines Knopfes des EnOcean-Schaltmoduls.

```
trigger:
  - platform: event
    event_type: button_pressed
    event_data:
        id: [0x00,0x00,0x00,0x00]
        pushed: 1
        which: 0
        onoff: 0
-   platform: state
    entity_id:  input_boolean.sleepmode_activation
    state: 'on'
```

Bei Auslösung eines der Trigger soll zunächst das Abspielen der Sounddatei starten. Das Abspielen findet über einen, in der `configuration.yaml` erstellten, Component statt.

```
shell_command:
    play_sound: /usr/bin/omxplayer /home/pi/sleep.wav
```

Dieser wird in der Automation aufgerufen.

```
action:
    - service: shell_command.play_sound
```

Aufgrund des starren Aufbaus der Automationen in Home Assistant ist die Nutzung von Schleifen nicht möglich. Die Umsetzung des simulierten Sonnenuntergangs lässt sich deshalb nur durch mehrere Zeilen Code lösen. Dazu wird ein Dimmer-Wert gesetzt und nach einer Verzögerung der nächste, kleinere Dimm-Schritt festgelegt. Dies ist eine sehr unschöne Lösung und ließe sich durch die Möglichkeit Schleifen in der Automation zu nutzen, einfach und mit weniger Zeilen Code lösen. In dem Szenario wird deshalb Anforderung IV nicht umgesetzt, sondern das Licht einfach ausgeschaltet.

```
-   service: light.turn_off
    data:
        entity_id: light.yeelight_rgb_286cxxxx
```

Nach einer festgelegten Zeit von 30 Minuten, bestimmt durch den Eintrag `delay: '00:30:00'` wird der Übergang in den Schlafmodus ausgelöst. Hierzu wird der Status eines weiteren Input-Boolean (sleep_mode_monitoring) angepasst und die Temperatur des Heizkörper-Thermostats auf 18 Grad Celsius gesetzt.

```
 - delay: '00:30:00'
 - service: input_boolean.turn_on
   data:
       entity_id: input_boolean.sleep_mode_monitoring
 - service: climate.set_temperature
   data:
       entity_id: climate.bad_thermostat_1
       temperature: 18
```

Anforderung VII:

Für die Schlafüberwachung dient die Änderung des CO2-Wertes oder der Zimmer-Temperatur als Trigger. Ändert sich der Wert und steigt dieser über einen festgelegten Grenzwert (definiert im Trigger durch den „above“-Wert), wird der Trigger ausgelöst.

```
trigger:
    -   platform: numeric_state
        entity_id: sensor.co2
        above: 600
    -   platform: numeric_state
        entity_id: sensor.temperature
        above: 21
```

Ist der Trigger aktiviert, wird überprüft, ob der Schlafmodus aktiv ist.

```
condition:
 -  condition: state
    entity_id: input_boolean.sleep_mode_monitoring
    state: 'on'
```

Ist dies der Fall, wird die automatische Belüftung gestartet.

```
action:
 -  service: switch.turn_on
    data:
    entity_id: switch.fan_control
```

Da Home Assistant ebenfalls nicht die Möglichkeit von if-else-Abfragen bietet, muss für das Ausschalten der Belüftung eine zweite Regel definiert werden, hier

allerdings mit dem Eintrag `below: 600`.

```
trigger:
    platform: numeric_state
    entity_id: sensor.co2
    below: 600
```

Für die Temperatur-Überwachung wird zusätzlich der Status des Fenstersensors überprüft, bevor die Belüftung gestartet wird.

```
  -  condition: state
     entity_id: sensor.fibaro_system_fgdw002_door_
                  opening_sensor_2_access_control
     state: '23'
```

Anforderung IX – XII:

Für die Aufweck-Automation wird ein Timer-Trigger genutzt (`platform: time`). Hier kann die Uhrzeit eingetragen werden.

```
  alias: 'WakeUp'
         trigger:
         -   platform: time
             at: '09:00:00'
```

Wird der Trigger abgefeuert, wird der „turn_on“-Service genutzt, um die Lampe einzuschalten, der Schlafmodus deaktiviert, die Temperatur des Thermostats auf 24 Grad gesetzt und die Sound-Datei abgespielt.

```
action:
  -  service: light.turn_on
     data:
         entity_id: light.yeelight_rgb_286cxxx
  -  service: input_boolean.turn_off
     data:
     entity_id: input_boolean.sleep_mode
  -  service: climate.set_temperature
     data:
         entity_id: climate.bad_thermostat_1
         temperature: 24
  -  service: shell_command.play_sound
```

Anforderung XIV:

Aufgrund zeitlicher Engpässe konnte die letzte Anforderung, das Aussetzen der

Aufweckfunktion, wenn der Nutzer nicht zu Hause ist, nicht mit in die Automation aufgenommen werden.

Das Einbinden der Lokalisierung mittels OwnTracks-App in die Smart Home Plattform funktionierte aber bereits problemlos. Dies legt die Vermutung nahe, dass auch diese Anforderung bei mehr Zeitreserven hätte umgesetzt werden können.

4.5.4 NodeRED

Installation

Seit November 2015 ist in der Raspberry Pi Distribution Raspbian Node-RED bereits vorinstalliert und sofort nutzbar. Da es sich hier allerdings um eine veraltete Version handelt, empfiehlt es sich Node-RED und Node.js zu aktualisieren. Dies ist durch folgenden Befehl möglich:

```
$ update-nodejs-and-nodered
```

Im Falle einer anderen Distribution oder einer alten Raspbian Version kann Node-RED auch manuell über den Package Manger installiert werden:

```
1 $ sudo apt-get update
  $ sudo apt-get install nodered
```

Gestartet wird Node-RED entweder über einen Doppelklick auf das Node-RED Icon im Startmenü oder durch den Kommandozeilenbefehl:

```
$ sudo node-red-start
```

Das automatische Starten von Node-RED beim Booten des Raspberry Pi wird mit folgendem Befehl festgelegt:

```
1 $ sudo systemctl enable nodered.service
```

Oberfläche

Nach der Installation ist die Node-RED Oberfläche im Browser unter der URL `http://ip-des-raspberry:1880` zu erreichen.

Die Oberfläche ist in drei Hauptbereiche unterteilt: der Node-Palette, dem Workspace und dem Informations- bzw. Debug-Panel.

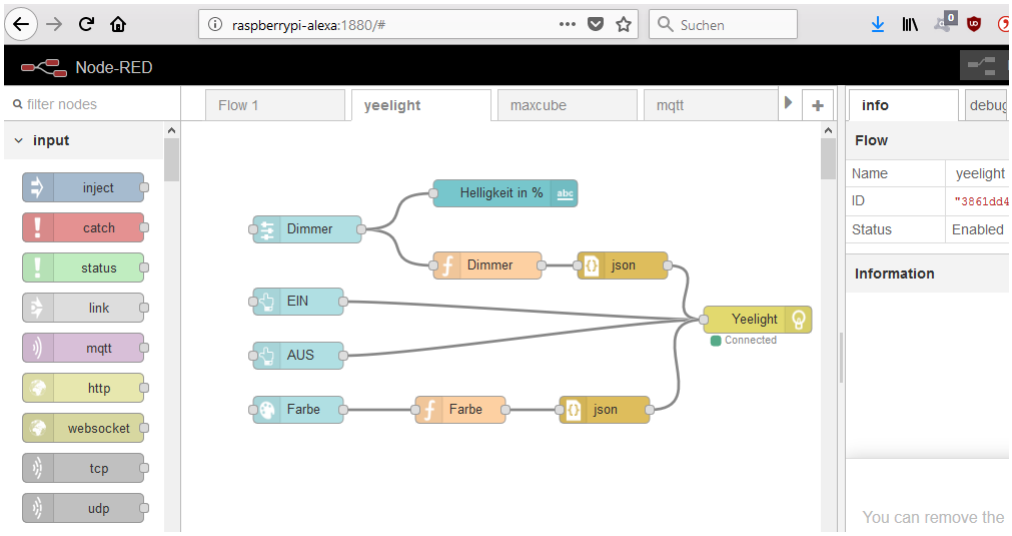


Abbildung 4.34: Node-RED Web-UI

Die Node-Palette befindet sich auf der linken Seite. Hier finden sich alle bereits installierten Nodes. Der Workspace erstreckt sich über den mittleren Bereich der Oberfläche. Hier werden flows definiert. Per Drag & Drop können die vorhandenen Nodes aus der Node-Palette in den Workspace gezogen, miteinander verbunden und konfiguriert und auf diese Weise Automationen definiert werden. Das Info-/Debug-Panel ist in weitere Tabs unterteilt. Im Informations-Tab werden Info-Texte zu den einzelnen Nodes und Anleitungen für die Konfiguration angezeigt. Der Debug-Tab zeigt die Ausgaben der Debug-Nodes und listet interne Fehler auf.

Konfiguration

In der Node-RED Grundkonfiguration sind bereits einige nützliche Nodes installiert. Werden für die Nutzung von Geräten und Diensten weitere Nodes benötigt, können diese direkt in der Web-Oberfläche nachinstalliert werden. Dazu muss der Menüpunkt *Manage Palette* im Drop Down Menü, welches sich rechts oben in der Weboberfläche befindet, ausgewählt werden. Hier kann nach neuen Nodes gesucht und diese per Mausklick installiert werden.

Für die Nutzung des **EnOcean**-Gateways bietet Node-RED vorgefertigte Nodes, die mit dem Paket „node-red-contrib-enocean“, über den Paletten-Manager nachinstalliert werden müssen. Insgesamt werden so drei neue Nodes installiert: die „enocean-listener“ Node, welche auf eingehende EnOcean-Telegramme wartet, sowie zwei EnOcean-Output-Nodes für die Steuerung von

Schaltern und Dimmern. Für das Smart Home Projekt wird die „enocean-listener“-Node benötigt. Für die Konfiguration muss der Pfad zu dem genutzten USB-Port angegeben werden.

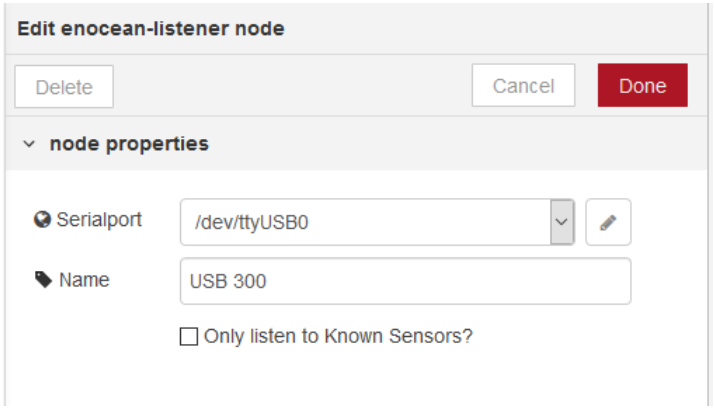


Abbildung 4.35: EnOcean-Node Konfiguration

Die Node basiert auf der „node-enocean-library“ von Node.js. Diese muss zusätzlich auf dem Raspberry Pi installiert werden.

```
1 npm install node-enocean
```

Nach einem Neustart ist die „enocean-listener“-Node in Node-RED verwendbar und das Gateway empfängt die Telegramme des Schalters.

Für die Nutzung von **Z-Wave** muss das „node-red-contrib-openzwave“-Paket installiert werden. Für die Konfiguration der benötigten Node muss wieder der genutzte USB-Port angegeben werden. Auch hier wird ein Node.js-Addon, na-

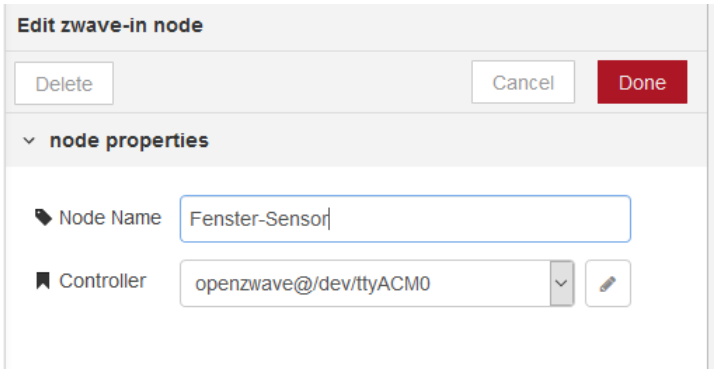


Abbildung 4.36: Z-Wave-Node Konfiguration

mens „node-openzwave-shared“, für die Verwendung in Node-RED benötigt.

Für die Installation dieses Addons sind einige weitere Installationsschritte nötig. Zunächst muss die libudev-Library installiert werden.

```
$ sudo apt-get install libudev-dev -y
```

Anschließend kann open-zwave von GitHub heruntergeladen und installiert werden.

```
$ git clone https://github.com/OpenZWave/open-zwave.git
$ cd open-zwave/
$ make
$ sudo make install
```

Zusätzlich muss noch openzwave-shared installiert werden.

```
$ cd ~/.node-red
$ npm cache clear
$ npm install openzwave-shared
```

Wenn open-zwave danach noch nicht funktioniert, muss die Environment-Variable aktualisiert werden.

```
$ export LD LIBRARY PATH=/usr/local/lib
```

Anschließend kann die Installation getestet werden. Hierzu muss Node.js gebootet

```
$ node
```

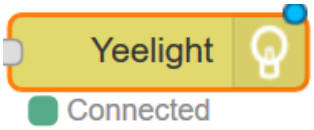
und eine, mit dem Paket mitgelieferte Datei zum Testen, gestartet werden.

```
> .load test2.js
```

Ein Fehler, der hier auftrat, war, dass in der Test-JavaScript-Datei ein anderer USB-Pfad als der genutzte angegeben ist. Hier musste zusätzlich in einem Editor der Pfad in der Test-Datei angepasst werden.

Findet dieses Test-Script den Z-Wave-Stick und empfängt es Statusänderungen von dem Z-Wave-Sensor, ist Z-Wave auch in Node-RED verwendbar.

Für die Nutzung der **Yeelight-LED** wird die „yeelight-compat-hue“-Node aus der Node-RED-Library installiert. Für die Konfiguration müssen die IP-Adresse der Yeelight sowie der genutzte Port (55443, zu entnehmen aus der Yeelight-Dokumentation) angegeben werden.



Für die Nutzung des **MAX! Cubes** bzw. des **Heizkörperthermostats**, gibt es vorgefertigte Nodes, die über den Palette Manager

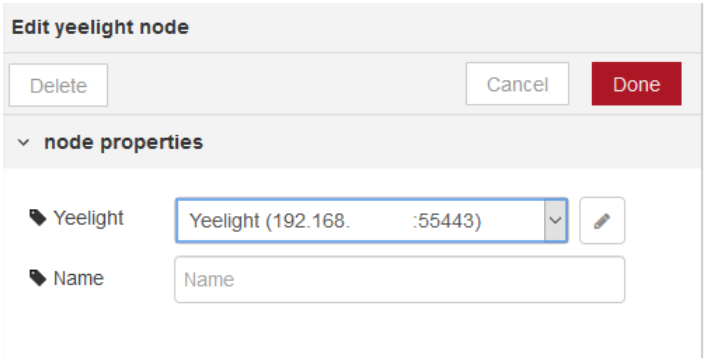


Abbildung 4.37: Yeelight-Node Konfiguration

installiert werden können. Für die Konfiguration müssen ebenfalls IP-Adresse und Port (62910) angegeben werden.

Für die Verwendung von MQTT ist bereits eine Node im Node-RED Standard-Paket vorhanden. In der Konfiguration müssen Server und Port des MQTT-Brokers sowie das zu abonnierende Topic angegeben werden.

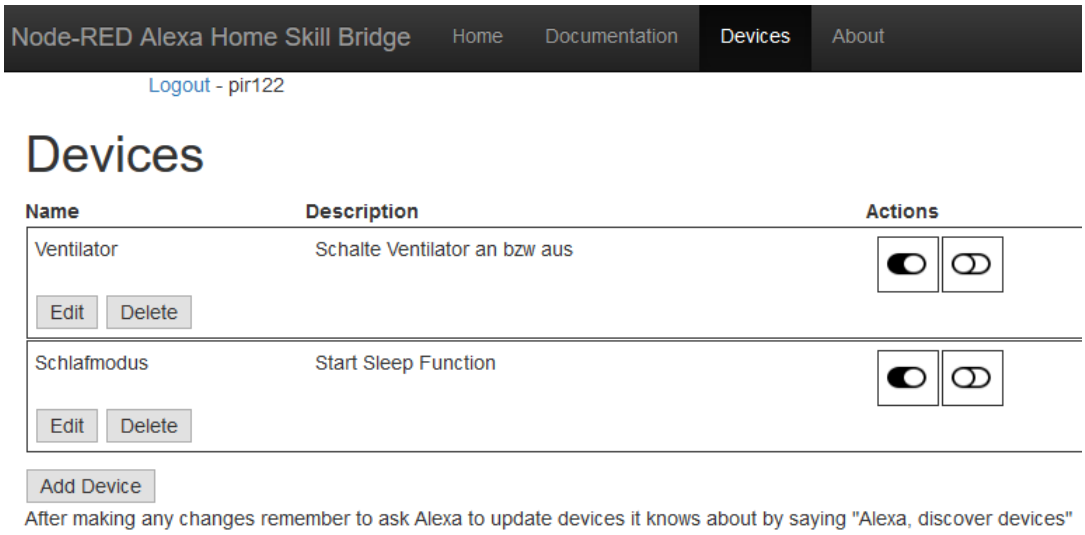


Abbildung 4.38: Node-RED Alexa Konto

Für die Nutzung des Alexa-Sprachassistenten muss ein Account erstellt werden unter:

https://alexa-node-red.bm.hardill.me.uk/newuser

Nach der Erstellung des Accounts müssen die Geräte definiert werden, welche gesteuert werden sollen. Ist man eingeloggt erscheint ein Tab mit dem Namen „Devices“. Unter diesem können mit dem Button „Add Devices“ die Geräte bestimmt werden, die über Alexa ansteuerbar sein sollen. Der hier vergebene Name ist der, der auch später in Alexa verwendet wird.

In Node-RED kann über den Palette-Manager die benötigte „alexa-home-skill“-Node nachinstalliert werden.

Für die Nutzung muss sich in der Node-Konfiguration über User-Namen und Passwort in den zuvor erstellten Account eingeloggt werden. Anschließend listet ein darunterliegendes Drop-Down Menü alle Devices, die zuvor in dem Online-Konto eingerichtet wurden. Über die Alexa App können diese Geräte gefunden und angesteuert werden.

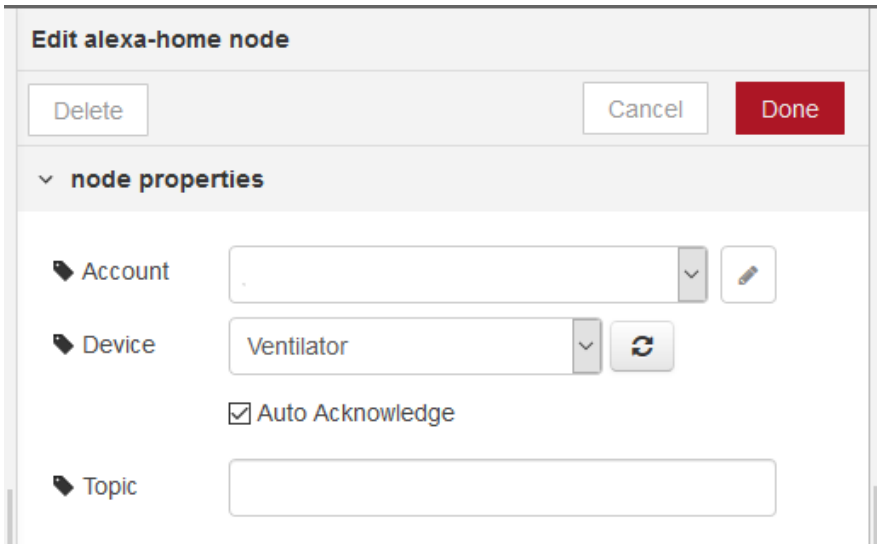


Abbildung 4.39: Node-RED Alexa-Node Konfiguration

Die Konfiguration der Own-Tracks-Node erwies sich als schwierig. Da Own-Tracks über MQTT kommuniziert, wurde die MQTT-Node verwendet, um die Daten in Node-RED zu empfangen. Das empfangene Paket muss über „function“-Nodes anschließend entsprechend gefiltert werden.

Für das im Projekt nicht genutzte ZigBee-Protokoll, wird von Node-RED eine Node zur Verfügung gestellt, die speziell für die Steuerung von smarten LED ausgelegt ist. Allerdings baut diese Node auch auf anderen Packages auf, die zuvor auf dem Raspberry Pi installiert und konfiguriert werden müssen.

Für die übrigen beiden Kommunikationsprotokolle, Bluetooth und HTTP, stellt Node-RED vorgefertigte Nodes zur Verfügung.

Visualisierung

Ein einfacher Weg, um eine ansprechende Bedienoberfläche in Node-RED zu erstellen, ist über das node-red-dashboard-Paket. Dieses muss über den Palette-Manager installiert werden. Es liefert eine Vielzahl von Nodes für unterschiedliche grafische Elementen wie Buttons, Slider, Texteingabefelder, Formulare, Textausgaben oder Diagramme. Ebenso ermöglicht es die Verwendung eigener HTML-Templates.

Aufgerufen wird die Bedienoberfläche mit der URL: `raspberrypi-ip:1880/ui`. Der Aufbau des User Interfaces kann direkt in den einzelnen Nodes oder über das Dashboard-Panel, welches als weiteres Tab im Info- /Debug-Panel angezeigt werden kann, vorgenommen werden. In der Konfiguration können die Elemente auf mehrere Tabs aufgeteilt und in Gruppen unterteilt werden. Gehören Elemente der gleichen Gruppe an, teilen sie sich eine Kachel in der UI. Je nach Art des Dashboard-Elements können Einstellungen wie x- und y-Achsen-Abschnitte oder Einheiten angepasst werden.

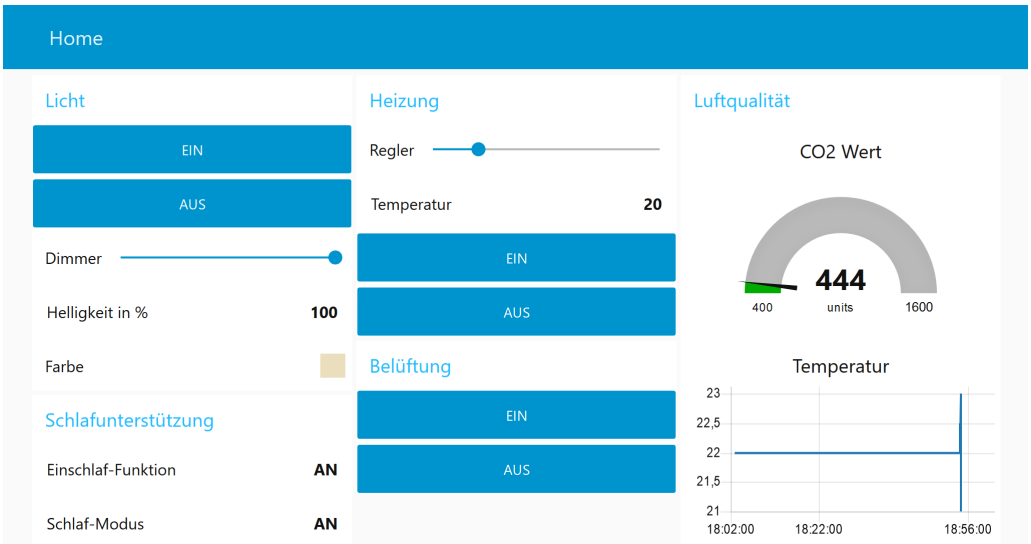
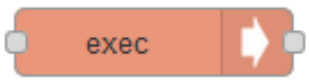


Abbildung 4.40: Node-RED Dashboard Web-UI

Erweiterbarkeit

Für das Ausführen von externen Scripten und Services des Raspberry Pi kann in Node-RED die „exec“-Node verwendet werden. Diese gehört zu den Standard-Nodes und muss nicht zusätzlich installiert werden. Nachdem die Node in den Workspace gezogen wurde, kann sie mit einem Doppelklick konfiguriert werden. Hier muss lediglich in das Feld „Command“ der Befehl eingetragen werden, der ausgeführt werden soll. Schon ist die Node fertig konfiguriert und kann beispielsweise für die Ansteuerung der Funksteckdose mittels des *raspberry-remote*-Services genutzt werden.



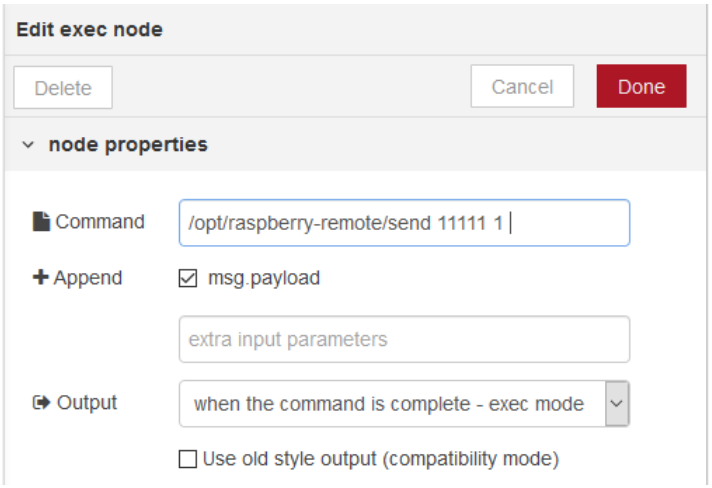


Abbildung 4.41: Node-RED Exec-Node Konfiguration

Nodes in Node-RED bestehen aus einem HTML-File, das für die Visualisierung der Node in der Web-Oberfläche zuständig ist und einer JavaScript-Datei, in welcher die Logik der neuen Node implementiert wird.

Für das Einbinden eigener Nodes muss ein neues Verzeichnis erstellt werden.

`/.node-red/nodes/beispiel-node`

In dieses werden die beiden Files für die neue Node eingefügt.

In der Node-RED Dokumentation finden sich Anleitungen und Tutorials zur Erstellung eigener Nodes.

Automation

Für Automationen müssen in Node-RED die einzelnen Nodes miteinander verbunden werden. Für die Kommunikation zwischen den einzelnen Nodes wird ein JavaScript-Objekt, das so genannte „msg“-Objekt, weitergeleitet. Der Body der Nachricht ist der so genannte „msg.payload“. In diesem steht die Information, die von einer Node zu der nächsten weitergereicht werden soll. Manche Nodes fügen noch zusätzliche Properties zu den msg-Objekten hinzu. So können weitere Informationen in dem msg-Objekt übertragen werden.

Ein wichtiges Hilfsmittel für die Erstellung von Automationen ist die „function“-Node. Diese erlaubt dem Nutzer kurze JavaScript-Codeblöcke zu schreiben und die msg-Objekte zu bearbeiten, bevor diese zur nächsten Node weitergeleitet werden.

Umsetzung Szenario

Anforderung I und II:

Für das Starten der Einschlaffunktion wird die konfigurierte „alexa-home“-Node in die Oberfläche gezogen. Diese versendet, wenn per Sprache aktiviert, ein msg-Objekt mit dem payload „true“. Ebenso wird die EnOcean-Listener Node in die Oberfläche gezogen. Diese sendet bei Betätigung eines der vier Buttons auf dem Schaltmodul einen, für jeden Schalter spezifischen, zweistelligen Code.

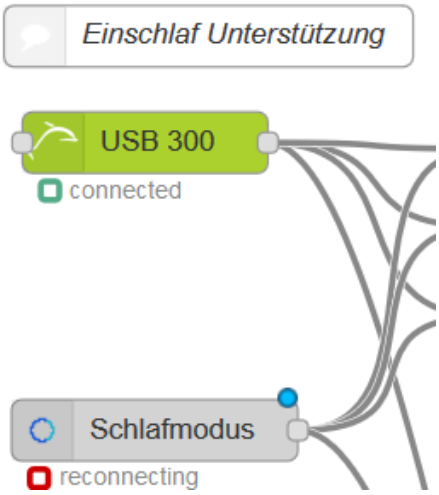


Abbildung 4.42: Node-RED Start der Einschlafunterstützung

Anforderung III:

Für das Abspielen der Sound-Datei wird die „play-audio-file“-Node verwendet. In dieser kann der Pfad zu einer Audio-Datei auf dem Raspberry-Pi angegeben werden, welche abgespielt wird sobald die Node ein msg-Objekt mit dem Payload „1“ empfängt. Zwischen der Sound-Node und der Alexa-Node bzw. der EnOcean-Node, wird eine „function“-Node gebaut. Diese überprüft mit einer If-Abfrage, ob Alexa gestartet, bzw. ein Schalter des Schaltmoduls betätigt wurde (Gewählt wurde der Schalter rechts oben, welcher mit dem msg-Objekt den Wert raw = „00“ versendet). Ist dies der Fall, wird der Payload des msg-Objektes mit dem Wert „1“ überschrieben und an die Audio-Node weitergeleitet, wodurch die Audiowiedergabe gestartet wird.

```
if(msg.payload == true || msg.payload.data.raw == "00")
{
    msg.payload = 1;
    return msg;
}
```

Abbildung 4.43: Node-RED Audio

Anforderung IV:

Für das Dimmen des Lichtes wird zwischen Alexa- bzw. EnOcean-Node eine „function“-Node gesetzt. In dieser wird eine globale Variable „dimVal“ definiert und ihr Wert auf 100 gesetzt. Der Wert wird für die Regelung der Helligkeit der Lampe genutzt. Eine weitere „function“-Node überprüft, ob der Wert der Variable noch positiv ist. Ist dies der Fall, schreibt sie den aktuellen Wert in den body des msg-Objektes und sendet den Wert zur Festlegung der Helligkeit an die Yeelight LED und verringert anschließend den Variablen-Wert.

```
if(context.global.dimmVal >= 0){
    msg.payload = {"bri":context.global.dimmVal}
    context.global.dimmVal --;
    return msg;
}
```

Nach einer kurzen Verzögerung wird die „function“-Node über eine Delay-Node erneut aufgerufen und mit dem neuen Wert der Variablen überprüft.

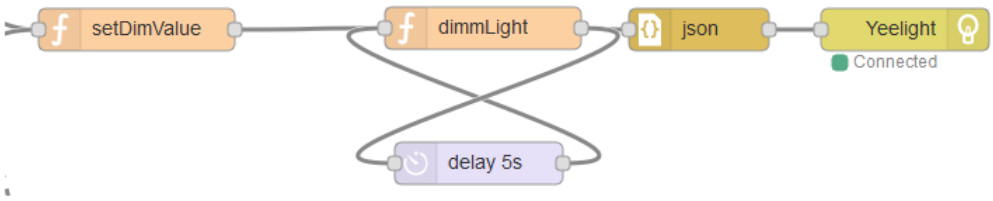


Abbildung 4.44: Node-RED Sonnenuntergang

Anforderung V:

Nach 30 Minuten wird der Schlafmodus aktiviert. Dafür wird eine Delay-Node eingesetzt, welche mit Alexa- bzw. EnOcean-Node verbunden ist. Eine globale Variable (sleepMode) wird mittels einer weiteren „function“-Node auf „true“ gesetzt.

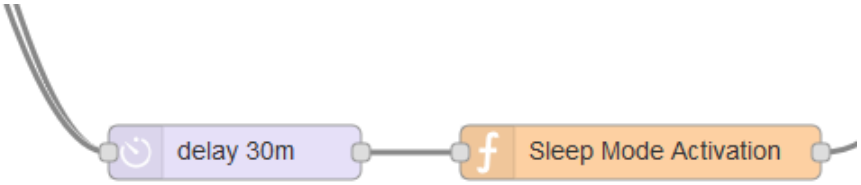


Abbildung 4.45: Node-RED Schlafmodus aktivieren

Anforderung VI:

Für die Regelung der Temperatur werden in den Body des msg-Objektes die RF-Adresse des Thermostats sowie die gewünschte Temperatur (18 Grad Celsius) eingetragen und an die MaxCube-Node weitergeleitet.

```
if(msg.payload == "AN"){
    msg.payload = {};
    msg.payload.rf_address = "14f733";
    msg.payload.degrees = 18;
}
return msg;
```

Anforderung VII:

Für die Überprüfung der Luftqualität bzw. der Temperatur wird das msg-Objekt der jeweiligen MQTT-Node an eine Switch-Node gesendet. Diese bietet, wie bei einer if-else-Abfrage, zwei Ausgänge. Ist der Grenzwert überschritten, wird der erste Ausgang gewählt und die Belüftung mittels exec-Node gestartet.

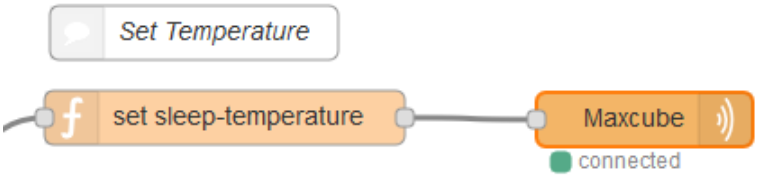


Abbildung 4.46: Node-RED Schlaftemperatur setzen

Unterschreitet der Wert die Grenze, verlässt das msg-Objekt die Node über den zweiten Ausgang und die Belüftung wird gestoppt.

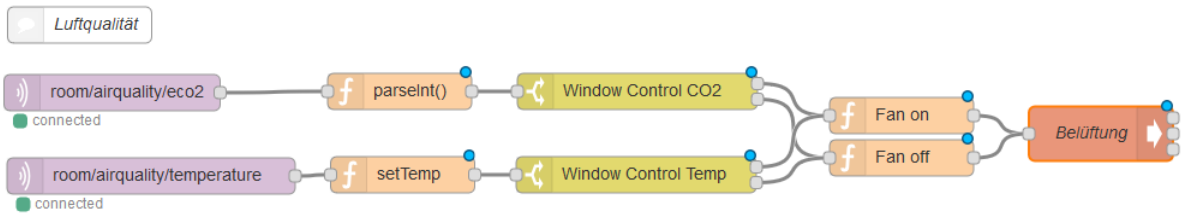


Abbildung 4.47: Node-RED Luftqualität

Anforderung VIII:

Für die Überprüfung des Fenster-Zustandes wird die Z-Wave-Node mit einer „function“-Node verknüpft. Diese passt eine globale Variable (Window-State) je nach Fenster-Status (geöffnet/geschlossen) an. Die Variable wird vor dem Starten der Belüftung, in einer if-else-Abfrage, überprüft.

Anforderung IX:

Für das Starten der Aufweckfunktion wird eine „inject“-Node verwendet. Diese gehört zu den Standard-Nodes und muss nicht zusätzlich installiert werden. In ihr kann festgelegt werden, zu welcher Uhrzeit ein zuvor festgelegter Payload versendet wird.

Anforderung X:

Für das Deaktivieren der Schlafüberwachung wird die globale Variable „sleep-Mode“ mittels JavaScript auf „false“ gesetzt.

Anforderung XI:

Das Setzen der Temperatur wird nach Erhalt des Startsignals durch die „inject“-Node wie in Anforderung VI durchgeführt.

Anforderung XII:

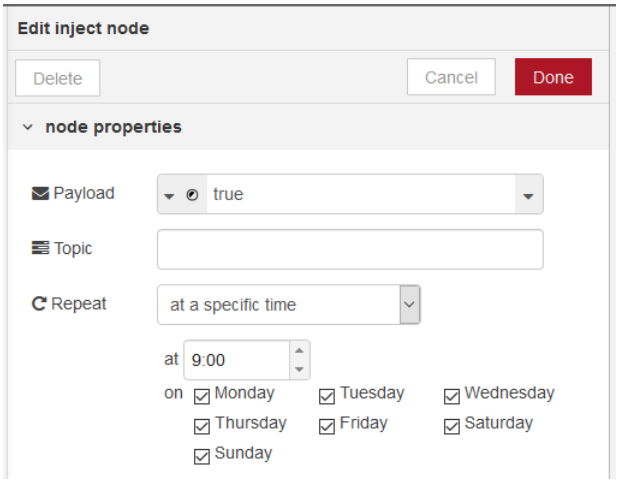


Abbildung 4.48: Node-RED Aufweckfunktion Uhrzeit Konfiguration

Das Simulieren des Sonnenaufgangs wird mit einer weiteren Variablen und durch das Heraufzählen des Dimmer-Wertes wie in Anforderung IV umgesetzt.

Anforderung XIII:

Das Abspielen der Sound-Datei erfolgt wie in Anforderung III.

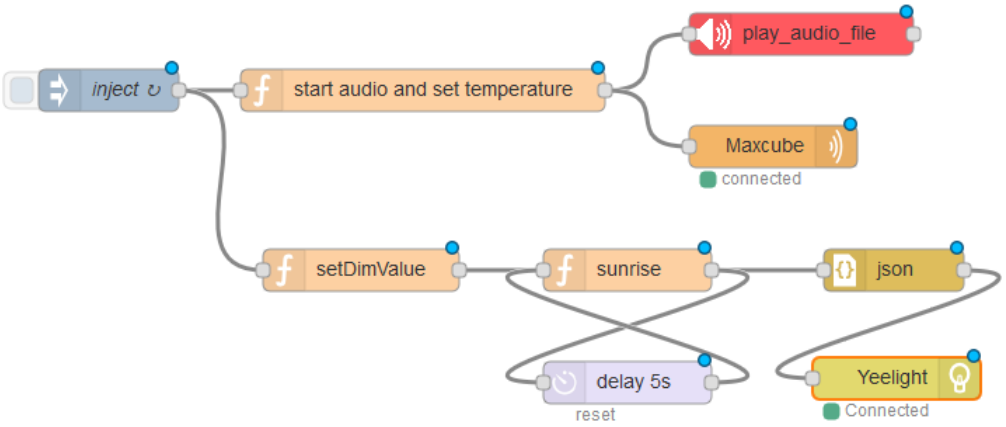


Abbildung 4.49: Node-RED Aufweckfunktion

Anforderung XIV:

Aufgrund zeitlicher Engpässe konnte die letzte Anforderung, das Aussetzen der Aufweckfunktion, wenn der Nutzer nicht Zuhause ist, nicht mit in die Automation aufgenommen werden.

Das Einbinden der GPS-Daten der OwnTracks-App in die Smart Home Platt-

form funktionierte aber bereits durch Abonnieren des entsprechenden MQTT-Topics in der MQTT-Node. Auch die Daten konnten bereits in einer „function“-Node weiterverarbeitet werden. Dies legt die Vermutung nahe, dass auch diese Anforderung bei mehr Zeitreserven, über den Umweg der MQTT-Node, hätte umgesetzt werden können.

4.6 Vergleich anhand der Vergleichskriterien

4.6.1 Installation

Bei jeder der vier Plattformen gestaltete sich die Installation problemlos. Für die manuelle Installation von ioBroker findet sich eine detaillierte Schritt-für-Schritt-Anleitung in der Online-Dokumentation. Einzig problematisch ist hier das eventuell notwendige „Downgraden“ der installierten Node.js-Version. Alternativ stellt ioBroker eine eigene Raspbian-Version bereit.

Bei OpenHAB ist ebenfalls positiv hervorzuheben, dass, neben der manuellen Installation, mit openHABian eine eigene Raspbian-Version bereitgestellt wird.

Home Assistant stellt mit Hass.io und Hassbian sogar mehrere Raspberry-Image-Versionen bereit, zwischen denen der Nutzer, je nach Anforderung, wählen kann.

Bei Node-RED gestaltete sich die Installation am einfachsten, da sie nicht nötig war. Wichtig ist hier Node-RED vor der Nutzung zu aktualisieren. Aber auch wenn Node-RED noch nicht bereits auf dem System installiert ist, kann dies mit nur zwei Befehlen nachgeholt werden.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Installation	+	+	+	++

4.6.2 Oberfläche

OpenHAB stellt dem Nutzer gleich mehrere Web-Oberflächen zur Verfügung. Die klassische Konfigurationsoberfläche Paper UI benötigt zunächst etwas Einarbeitungszeit. Aber nach einer Weile erkennt man die klare OpenHAB-Strukturierung nach Things, Bindings, Channels und Items. Das User Interface ist modern und ansprechend gestaltet und bietet viele Einstellmöglichkeiten.

Die Web-Oberfläche von ioBroker ist klar strukturiert und in Adapter, Instanzen und Objekte untergliedert. Die Konfiguration der einzelnen Geräte und Dienste kann vollständig in der Web-Oberfläche durchgeführt werden. Die Gestaltung der Weboberfläche wirkt dabei allerdings eher altbacken und auch die Ordnerstruktur der Objekte und Scripte wirkt etwas verworren.

Home Assistant stellt eine moderne Web-Oberfläche bereit. Die einzelnen Funktionen sind allerdings teilweise etwas versteckt und müssen zunächst gesucht werden, was die Nutzung weniger intuitiv macht. Deshalb dauerte die Einarbeitung in die Home Assistant Oberfläche länger, als bei den anderen Smart Home Plattformen. Allerdings stellt Home Assistant in seiner Web-Oberfläche auch eine Vielzahl von Funktionalitäten bereit, wie beispielsweise den Automations-Editor.

Node-RED verfolgt für die Darstellung seiner Web-Oberfläche einen komplett anderen Ansatz als die übrigen Smart Home Plattformen. Auf diese Weise stellt Node-RED auch die ansprechendste Web-Oberfläche bereit. Die Nutzung ist durch die klare Strukturierung der Oberfläche sehr intuitiv. Sie ist sehr übersichtlich gehalten und wirkt aufgeräumt. Der grafische Aufbau der Automationen, in Form von miteinander verbundenen Blöcken, ist ansprechend und logisch. Info-Texte zu den einzelnen Nodes werden direkt in der Oberfläche angezeigt.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Oberfläche	+	O	O	++

4.6.3 Konfiguration - Anzahl integrierbarer Technologien

Die Konfiguration der benötigten Geräte und Dienste ließ sich in fast allen Plattformen vollständig umsetzen. Lediglich bei ioBroker wurde ein verwendbarer Adapter für EnOcean vermisst. Ebenso konnte die Nutzung der Yeelight LED nur über einen Umweg ermöglicht werden. Auch für, im Smart Home Szenario nicht genutzte Protokolle, wie ZigBee und HTTP, stellte ioBroker keine eigenen Adapter bereit.

OpenHAB stellt ebenso kein Binding für die Nutzung der Yeelight LED bereit. Dieses konnte allerdings von einer externen Quelle hinzugefügt werden.

In Home Assistant und Node-RED konnten alle benötigten Components bzw. Nodes eingebunden und konfiguriert werden.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Technologien	+	O	++	++

4.6.4 Konfiguration - Einfachheit der Umsetzung

Die Konfiguration in OpenHAB gestaltete sich unkompliziert. Das textuelle Anlegen der einzelnen Items hatte allerdings seine Tücken. Zunächst muss ein neues Verzeichnis für die Items händisch angelegt werden. Die Syntax der Items muss genau eingehalten werden und teilweise waren sehr lange Textblöcke für das Definieren eines Items nötig. Hier können schnell Eingabefehler passieren. Die Entwickler von OpenHAB scheinen dieses Problem auch erkannt zu haben und neuere Bindings können nun komplett im Web-Editor, inklusive Item und ohne textuelle Angaben, angelegt werden.

Bei ioBroker findet die Konfiguration der smarten Geräte und Dienste komplett in der Weboberfläche statt. So läuft man nicht Gefahr, dass irgendwelche Syntax-Fehler entstehen. Die Konfiguration wird direkt an dem jeweiligen Adapter in einem separaten Fenster durchgeführt, das sich bei einem Klick auf den Adapter öffnet.

Noch einfacher gestaltete sich die Konfiguration in Home Assistant. Die Yeelight LED wurde beispielsweise schon beim ersten Start von Home Assistant automatisch eingebunden und war sofort verwendbar. Die Konfiguration in Home Assistant ist zwar auch textuell, allerdings geschieht das Einbinden der Geräte hauptsächlich mit Copy & Paste und dem anschließenden Anpassen der Key-Value-Paare. Ausführliche Anleitungen für die Konfiguration der Components unterstützen den Nutzer dabei. So konnte die Konfiguration in Home Assistant auch am schnellsten durchgeführt werden.

Am schwierigsten gestaltete sich die Konfiguration bei Node-RED. Das Einrichten der Nodes direkt in der Weboberfläche ist intuitiv und die Konfiguration klar aufgebaut. Die Info-Texte im rechten Panel sind eine gute Hilfestellung bei der Einrichtung der Nodes. Allerdings merkte man bei der Konfiguration typischer Smart Home-Protokolle, wie EnOcean oder Z-Wave, dass Node-RED doch viel mehr eine visuelle Programmierumgebung für das Internet der Dinge und weniger eine Smart Home Plattform, darstellt. So gestaltete sich etwa die Konfiguration der Z-Wave-Node als durchaus kompliziert, da hier für die Nutzung noch zusätzliche Software auf dem Raspberry Pi installiert und eingerichtet werden musste. Der Z-Wave-Adapter von ioBroker basiert beispielsweise

auf dem gleichen Node.js-Addon, wie die Node-RED-Node. Allerdings wird hier dem Nutzer die Installation der zusätzlichen Pakete abgenommen.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Einfachheit	O	+	++	-

4.6.5 Visualisierung

In OpenHAB lässt sich mit wenigen Schritten eine optisch ansprechende Bedienoberfläche einrichten. Diese muss dabei allerdings textuell definiert werden, was die Erstellung im Vergleich zu anderen Smart Home Plattformen etwas komplizierter macht. Von Beginn an hat man in OpenHAB die Möglichkeit, zwischen zwei Bedienoberflächen zu wählen und diese durch den Einsatz von Icons noch etwas den eigenen Wünschen anzupassen. Zusätzlich bietet das HABPanel-Dashboard, das ohne großen Aufwand über die Paper UI installiert werden kann, die Möglichkeit, per Drag & Drop in kürzester Zeit eine alternative Bedienoberfläche in vollkommen anderem Design einzurichten. Diese kann noch weiter den individuellen Wünschen angepasst werden, beispielsweise durch das Einbinden von eigenen Bildern in die UI.

Die Visualisierung ist die Paradedisziplin von ioBroker. Speziell der Vis-Adapter sticht im Vergleich der Smart Home Plattformen als innovative Möglichkeit, auf einfachem Weg eine komplett individuelle Bedienoberfläche zu implementieren, heraus. Das Erstellen des User Interfaces ist intuitiv. Die Elemente werden per Drag & Drop in die Oberfläche gezogen und können dort direkt den Geräten und Diensten zugewiesen werden. Durch die Möglichkeit, eigene Bilder direkt in die Editor-Oberfläche zu ziehen oder die Bedienoberfläche mit der Stylesheet-Sprache CSS pixelgenau zu definieren, kann eine, auf die eigenen Wünsche zugeschnittene, Benutzerschnittstelle eingerichtet werden. Einziger Kritikpunkt ist, dass die Auflösung der Oberfläche zuvor festgelegt werden muss. Eine responsive Darstellung wird nicht unterstützt. Allerdings können mehrere Screens in unterschiedlicher Auflösung für die genutzten Geräte angelegt werden. Meist wird in einem Smart Home nur ein einziges Device, beispielsweise ein an der Wand angebrachtes Tablet, als Steuerzentrale verwendet. Dies relativiert den Kritikpunkt etwas.

Die Standard-Bedienoberfläche von Home Assistant ist modern und optisch ansprechend. Allerdings bietet diese kaum Möglichkeiten für eine individuelle

Darstellung. Die Optionen beschränken sich auf Kleinigkeiten, wie das Ausblenden einzelner Elemente oder die Anpassung der Farbe. Die Installation des Home Assistant Dashboards ermöglicht das Einrichten einer alternativen Bedienoberfläche. Allerdings gestaltet sich die Installation etwas aufwändiger.

In Node-RED kann in sehr kurzer Zeit ein optisch ansprechendes Dashboard erstellt werden. Bei der Darstellung der Elemente ist man allerdings auf die Vorgaben der Nodes beschränkt. Das Einrichten einer individuellen Bedienoberfläche nach den eigenen Wünschen, wird hier, ähnlich stark wie bei Home Assistant, eingeschränkt.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Visualisierung	+	++	O	O

4.6.6 Erweiterbarkeit

Die Erweiterbarkeit ist bei allen Smart Home Plattformen vollständig zufriedenstellend. Dies ist auch nicht weiter verwunderlich, da Open Source Software davon lebt, dass eine breite Community die Software ständig weiterentwickelt. Interessant war es hier zu untersuchen, wie kompliziert es ist, das System zu erweitern.

In allen Plattformen bestand die Möglichkeit externe, Scripte direkt aus der Oberfläche zu starten. Dies stellt die einfachste Art, das System durch zusätzliche Services zu erweitern, dar. Für das Schreiben eigener Erweiterungen stellt jeder der Plattform-Anbieter ausführliche Anleitungen und Tutorials bereit.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Erweiterbarkeit	++	++	++	++

4.6.7 Automation

Die Erstellung von intelligenten Abläufen ist eine Stärke von OpenHAB. Der Aufbau einer Regel ist logisch nach Auslöser und Script Block unterteilt. Die

Möglichkeit, in den Regeln auch programmiersprachen-typische Kontrollstrukturen, wie Schleifen, Bedingte Anweisungen und Verzweigungen zu nutzen, erlaubt die Erstellung von komplexen Programmabläufen.

An der Automation erkennt man, dass es sich bei Node-RED nicht um eine wirkliche Smart Home Plattform handelt. Die anderen Software-Plattformen gehen bei der Konfiguration nach einem ähnlichen Schema vor. Für jedes Gerät und jeden Dienst wird eine Art Objekt angelegt. Diese Objekte haben einen bestimmten Status oder Zustand und besitzen Datenpunkte. So können in diesen Plattformen auch Trigger angelegt werden, die bei der Änderung eines Objektes ausgelöst werden. Dies ist bei Node-RED nicht möglich. Der objektorientierte Aufbau der anderen Smart Home Plattformen fehlt bei Node-RED. Die Geräte und Dienste können über die Nodes angesprochen und ihre Werte ausgelesen werden. Allerdings arbeitet hier jede Node für sich. Die Kommunikation findet nur über die, zwischen den Nodes versendeten, msg-Objekte statt. So können zusammenhängende Abläufe auch nicht auf mehrere Flows aufgeteilt werden, sondern müssen in einem langen, aneinanderhängenden Flow definiert werden. Dies kann, gerade bei komplexeren Abläufen, zu sehr langen und deshalb unübersichtlichen Programmen führen.

Dabei bieten sich die Flows für Darstellung von intelligenten Abläufen grundsätzlich sehr gut an. Auf diese Weise können die Abläufe logisch strukturiert und visualisiert werden. Dies könnte auch einer der Gründe sein, warum in ioBroker zusätzlich die Möglichkeit besteht, Node-RED für die Erstellung von Abläufen zu nutzen. Hier werden allerdings die in der ioBroker Oberfläche eingerichteten Objekte mit einer ioBroker-Node angesprochen. Die zusätzliche Möglichkeit, Abläufe auch direkt mit JavaScript zu implementieren, ist ein mächtiges Werkzeug für das Erstellen von komplexeren Abläufen. IoBroker macht hier vieles richtig und gibt dem Nutzer die Wahl, auf welche Weise er die Abläufe implementieren möchte. So kann je nach Programmier-Erfahrung des Nutzers und Komplexität der Anforderung, die passende Möglichkeit ausgewählt werden.

Die Erstellung von Automationen in Home Assistant ist grundsätzlich sehr logisch aufgebaut. Nach einer gewissen Einarbeitungszeit in die Auszeichnungssprache YAML und deren Syntax können schnell einfache Automationen erstellt werden. Die fehlende Möglichkeit, Schleifen und Bedingte Anweisungen, wie If-Abfragen in den Aktionen einzusetzen, schränkt den Nutzer bei der Erstellung komplexerer Abläufe stark ein. Diese können oft nur umständlich oder durch unnötig viele Zeilen an Code gelöst werden.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Automation	++	++	O	O

4.6.8 Verbreitung

Nach der Umsetzung des Smart Home Szenarios und der damit intensiveren Beschäftigung mit den Software-Plattformen, ist es interessant noch einmal auf die Verbreitung der vier Open Source Lösungen einzugehen.

Alle Plattformen bieten eine große Sammlung an Informationen, sei es in Form von ausführlichen Dokumentationen, Hilfsforen oder Tutorials. Über die größte Community verfügt Home Assistant. Eine ausführliche Dokumentation und eine Vielzahl von Beispielen erleichtern dem Nutzer den Einstieg in die Plattform. Auch die Anzahl der für die Plattform bereits vorhandenen Geräte und Dienste ist, verglichen mit den anderen, riesig.

Aber auch hinter OpenHAB steht eine sehr große und aktive Community und eine ausführliche Dokumentation wird bereitgestellt, auch wenn diese teilweise etwas veraltet ist bzw. vieles noch für die frühere Version OpenHAB 1.x erstellt wurde. Zu jedem Problem, das während der Umsetzung des Smart Home Szenarios auftrat, wurden oft sogar mehrere Lösungs-Ansätze in den verschiedenen OpenHAB-Foren angeboten und es gab eigentlich nichts, was nicht zuvor schon einmal in einem Forum diskutiert wurde.

Node-RED stellt die anderen Open Source Plattformen, was die Anzahl der Google-Sucherergebnisse betrifft, in den Schatten. Allerdings beschäftigt sich, wie bereits in 3.2.4 vermutet, der Großteil der Ergebnisse nicht mit Themen aus dem Smart Home Bereich. So gestaltete sich die Suche nach Problemlösungen im Smart Home Szenario auch meist schwerer als bei Home Assistant oder OpenHAB bzw. es wurden weniger Ergebnisse zu speziellen Smart Home Problemen gefunden.

Die kleinste Community ist bei ioBroker zu finden. Hier wurden bei der Suche nach Problemlösungen die wenigsten Ergebnisse gefunden. Dies könnte daran liegen, dass es sich hauptsächlich um eine deutschsprachige Community handelt. Die Dokumentation im Bereich Smart Home ist aber auch bei ioBroker vorbildlich. Die ioBroker-Community ist zwar klein, aber sehr aktiv. So wurde auch hier zu jedem Problem, das im Laufe der Umsetzung des Smart Home Szenarios auftrat, in Foren mindestens eine Lösung oder Erklärung gefunden. Mit den anderen Plattformen kann ioBroker allerdings trotzdem nicht mithal-

ten, was die Anzahl der Tutorials, Foren-Einträgen und die Ausführlichkeit der Dokumentation angeht.

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Verbreitung	+	-	++	O

4.6.9 Gesamttabelle

	OpenHAB	**ioBroker**	**Home Assistant**	**Node-RED**
Installation	+	+	+	++
Oberfläche	+	O	O	++
Technologien	+	-	++	++
Einfachheit	O	+	++	-
Visualisierung	+	++	O	O
Erweiterbarkeit	++	++	++	++
Automation	++	++	O	O
Verbreitung	+	-	++	O

5 Schluss

5.1 Zusammenfassung

Ziel dieser Bachelor-Thesis war es, sich mit den vier Open Source Smart Home Plattformen OpenHAB, ioBroker, Home Assistant und Node-RED auseinanderzusetzen, Vergleichskriterien zu entwickeln und die Plattformen in einem exemplarischen Smart Home Szenario miteinander zu vergleichen.

Um ein Grundverständnis von dem Begriff Smart Home zu entwickeln und diesen breitgefächerten Begriff genauer einzugrenzen, wurde in Kapitel 2 zunächst auf die Begriffe Internet der Dinge und Smart Home genauer eingegangen und eine Definition für diese ausgearbeitet. Zusätzlich sollten noch weitere Fragen geklärt werden: Wie ist die historische Entwicklung des Smart Homes? Seit wann existiert der Begriff Smart Home? Wie sieht die grundsätzliche Architektur eines Smart Homes aus und wo finden sich Anwendungsgebiete? Des Weiteren wurde nach den wichtigsten Smart Home Technologien recherchiert, wie die meistgenutzten Kommunikationsstandards und die Anbindung von Cloud-Diensten. Beispiele von bereits umgesetzten Smart Homes sollten dabei helfen, den Begriff noch besser zu verstehen. Diese Recherche sollte die Grundlage für die spätere Erstellung eines exemplarischen Smart Home Szenarios bieten und dabei helfen, ein möglichst realitätsnahes Szenario zu entwickeln, das möglichst viele unterschiedliche Gesichtspunkte eines intelligenten Zuhauses abbildet.

In Kapitel 3 wurde der Open Source Begriff genauer definiert, um anschließend die vier zu vergleichenden Open Source Plattformen genauer zu untersuchen. Betrachtet wurden dabei speziell der Funktionsumfang der Plattform, die Verbreitung und die Lizenz. Zusätzlich wurden Besonderheiten der Smart Home Lösungen gesammelt. So konnte sich ein erster Überblick über die Plattformen, ihre Möglichkeiten und ihren Funktionsumfang verschafft werden.

Für den konkreten Vergleich wurde in Kapitel 4 ein exemplarisches Smart Home Projekt entworfen, welches anschließend verwirklicht werden sollte. Die Entscheidung fiel auf die Umsetzung eines Smart Home Schlafassistenten, welcher den Nutzer beim Einschlafen und dem Aufwachen unterstützt und zu einem angenehmen Schlaf beitragen soll. Dafür wurden zunächst die Anforderungen an

das exemplarische Smart Home Szenario definiert, welche sich unter anderem aus den zuvor recherchierten Kommunikationsstandards ergaben. Diese Anforderungen wurden anschließend anhand ihrer Bedeutung für ein möglichst realitätsnahes Smart Home Szenario priorisiert. Anschließend wurden Vergleichskriterien definiert, anhand derer der spätere Vergleich der unterschiedlichen Smart Home Plattformen stattfinden sollte. Für den anschließenden Vergleich wurden die Vergleichskriterien tabellarisch aufgelistet und die Smart Home Plattformen in jeder der Kategorien bewertet.

5.2 Fazit

Der Vergleich der vier Open Source Smart Home Plattformen zeigt, dass jede Software-Lösung ihre Vorteile bietet und die Macher der jeweiligen Lösung sich intensive Gedanken bei der Implementierung gemacht haben. Mit jeder der vier Plattformen war die Umsetzung des Smart Home Szenarios (weitestgehend) möglich. Teilweise wählen die Software-Lösungen sehr unterschiedliche Lösungsansätze für die Implementierung eines Smart Homes. Die Eignung für den Einsatz im Smart Home konnten alle Plattformen nachweisen. Komplett abgeraten werden kann deshalb von keiner. Es muss eher nach Präferenz bzw. Anwender unterschieden werden, welche Plattform die „beste“ Smart Home Lösung darstellt.

Ein wichtiger Faktor ist die Größe der Community bzw. die Verbreitung der Plattform. Anhand dieser kann zusätzlich eine Prognose zur Zukunftsfähigkeit des jeweiligen Projektes abgegeben werden. Hier sticht vor allem Home Assistant aufgrund seiner Verbreitung und der riesigen Zahl an bereits verfügbaren Geräten und Diensten hervor. Es stellt damit die größte der verglichenen Plattformen (im Bereich Smart Home) dar. Diesen Spitzenplatz wird Home Assistant, was die Aktualität der Foreneinträge und der nutzbaren Geräte vermuten lässt, weiterhin halten und sogar noch weiter ausbauen. Ein Schwachpunkt von Home Assistant ist die Automation. Im exemplarischen Smart Home Projekt wurde die Möglichkeit, Schleifen und Bedingte Anweisungen nutzen zu können, vermisst. Aber für Nutzer ohne Programmierkenntnisse dürfte der simple Aufbau der Automationen in Home Assistant einen klaren Vorteil darstellen. Speziell die noch relativ junge Hass.io-Distribution von Home Assistant stellt durch den kompletten Verzicht auf den Einsatz der Kommandozeile, einen interessanten Ansatz dar, um mit der Plattform den so genannten „Mainstream“ anzusprechen. So stellt diese Plattform mit ihrer Einfachheit und Größe für den normalen Nutzer bzw. den Nicht-Entwickler die vielleicht interessanteste Lösung dar.

Der Exot unter den verglichenen Smart Home Plattformen ist ioBroker. Dieser hat die kleinste Community und auch noch am wenigsten verfügbare Geräte. Allerdings hat ioBroker eine sehr aktive, deutschsprachige Community und speziell der Vis-Adapter, für die Erstellung von ansprechenden Bedienoberflächen, aber auch die verschiedenen Optionen für die Erstellung von intelligenten Abläufen, machen die Plattform interessant. Deshalb ist es für (zukünftige) Nutzer eines Smart Homes durchaus empfehlenswert, sich auch mit dieser Plattform zu beschäftigen. IoBroker kann dabei für Leute interessant sein, die Wert auf eine ansprechende Visualisierung legen, tiefer in die Materie einsteigen wollen und die dabei eventuell auch dazu bereit sind, eigene Adapter für die Einbindung neuer Geräte zu schreiben.

Node-RED hat im Bereich des Smart Homes am schwächsten abgeschnitten. Die Einrichtung der Geräte war mit dieser Plattform am kompliziertesten. Allerdings ließen sich auch mit dieser alle Anforderungen des exemplarischen Smart Home Szenarios umsetzen. Node-RED kann vor allem für Nutzer interessant sein, welche bereits Projekte mit dem Raspberry Pi umgesetzt haben. Eine Node-RED Installation ist bei Raspbian meistens bereits vorhanden. So kann Node-RED auch ohne großen Konfigurations-Aufwand direkt eingesetzt werden und ist am schnellsten verwendbar. Aufgrund seiner Komplexität richtet sich Node-RED allerdings eher an Entwickler, mit zumindest grundlegenden Programmier- und Linux-Kenntnissen und nicht an die breite Masse, was die anderen Plattformen immer mehr versuchen.

Den vielleicht besten Kompromiss aus Größe und Komplexität stellt für technisch Interessierte OpenHAB dar. OpenHAB konnte in keiner der Vergleichskategorien gewinnen, wusste aber trotzdem in jeder der Kategorien zu gefallen. Hinter dieser Plattform steht eine große und aktive Community, wenn auch nicht so groß wie die von Home Assistant. Eine ansprechende Bedienoberfläche ist einfach umzusetzen. Die Konfiguration der Geräte gestaltete sich etwas anspruchsvoller als bei anderen Smart Home Lösungen, war aber auch nicht zu kompliziert. Die Macher von OpenHAB arbeiten auch intensiv daran, diese noch weiter zu vereinfachen. Dies zeigt auch die Tatsache, dass bei neueren Bindings versucht wird, auf die textuelle Einrichtung zu verzichten. Aufgrund seiner Nähe zu Java lassen sich bei OpenHAB auch komplexe Automationen gut umsetzen. So wird auch die Wahl des Verfassers dieser Thesis für die Umsetzung zukünftiger privater Smart Home Projekte voraussichtlich auf OpenHAB fallen.

Literaturverzeichnis

[1] F. Scholl, “Can a virtual marketing agency manage my company’s website?” 2016. [Online]. Available: http://www.rhinopros.com/blog/can-a-virtual-marketing-agency-manage-my-companys-website

[2] K. Rixecker, “Nerd-galerie: So sah ein smart-home-system im jahr 1990 aus,” 2014. [Online]. Available: http://t3n.de/news/retro-smart-home-system-548431/

[3] C. Pätz, *Z-Wave - Die Funktechnologie für das Smart Home*, 3rd ed. Norderstedt: BoD - Books on Demand, April 2017.

[4] R. Glasberg, “Studienreihe zur heimvernetzung: Konsumentennutzen und persönlicher komfort.”

[5] C. Baun, *Computernetze kompakt*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[6] HiveMQ, “Mqtt 101 – how to get started with the lightweight iot protocol,” 2015. [Online]. Available: https://www.hivemq.com/blog/how-to-get-started-with-mqtt

[7] N. Jurran, “Schlaues heim, glück allein: Smart home in der praxis,” *c’t*, no. 7, pp. 110–117.

[8] Universität Bremen, “Baall - bremen ambient assisted living lab,” 02.02.2012. [Online]. Available: http://baall-www.informatik.uni-bremen.de/de/index.php/Hauptseite

[9] MagPi, “Tierische projekte,” p. 90, 05.2016.

[10] E. Tsai, “Dog tracker,” 2014. [Online]. Available: https://www.hackster.io/erictsai/uber-home-automation-fd5a0f/logs/4

[11] Reichelt Elektronik, “Raspberry pi,” 2018. [Online]. Available: https://www.reichelt.de/?ARTICLE=152728

[12] art of circuits, “Esp32 dual core wifi+bluetooth development board,” 2018. [Online]. Available: http://artofcircuits.com/product/banana-pi-1g-ddr3-ram-1ghz-arm7-dual-core-processor-embedded-board

[13] EnOcean GmbH, "Ptm 210 schaltmodul," 2018. [Online]. Available: https://www.enocean.com/de/enocean_module/ptm-210/

[14] Amazon, "Z-wave kontaktsensor," 2018. [Online]. Available: https://www.amazon.de/Fibaro-T%C3%BCr-Fensterkontakt-GEN5-FIBEFGK-101-ZW5/dp/B01GCPKON4

[15] elv, "Maxcube thermostat," 2018. [Online]. Available: https://www.elv.de/elv-max-set-mit-1x-cube-und-1x-heizkoerperthermostat.html

[16] Amazon, "Yeelight smart led bulb," 2018. [Online]. Available: https://www.amazon.com/YEELIGHT-Equivalent-Smartphone-Controlled-Assistant/dp/B01LRTWQNG

[17] MiniInTheBox, "433mhz funk-sendemodul," 2018. [Online]. Available: https://www.miniinthebox.com/de/433mhz-funk-sendemodul-superregenerationsschaltung-fuer-arduino-gruen_p411875.html

[18] A. Cockburn, "Dashboard install and configuration," 2018. [Online]. Available: http://appdaemon.readthedocs.io/en/stable/DASHBOARD_INSTALL.html

[19] A. Nordrum, "Popular internet of things forecast of 50 billion devices by 2020 is outdated," 2016. [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[20] R. van der Meulen, "Gartner says 8.4 billion connected "things" will be in use in 2017," 2017. [Online]. Available: https://www.gartner.com/newsroom/id/3598917

[21] R. J. Vetter, "Internet kiosk- computer-controlled devices reach the internet," *Computer*, vol. 28, no. 12, p. 66, 1995.

[22] "Cmu scs coke machine." [Online]. Available: http://www.cs.cmu.edu/~coke/

[23] A. Gabbai, "Kevin ashton describes "the internet of things"," 2015. [Online]. Available: https://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/

[24] K. Ashton, "That 'internet of things' thing," 2009. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[25] S. Horvath, "Aktueller begriff internet der dinge," 2012. [Online]. Available: https://www.bundestag.de/blob/192512/.../internet_der_

dinge-data.pdf

[26] D. Evans, "The internet of things how the next evolution of the internet is changing everything," 2011.

[27] V. M. P. G. Pflegerl, J., *passgenau helfen.* Lit-Verlag, 2013. [Online]. Available: https://books.google.de/books?id=TiSEXvzzUaMC

[28] S. U. K. T. B. A. Strese, Hartmut, "Smart home in deutschland: Untersuchung im rahmen der wissenschaftlichen begleitung zum programm next generation media (ngm) des bundesministeriums für wirtschaft und technologie," 2010. [Online]. Available: https://www.iit-berlin.de/de/publikationen/smart-home-in-deutschland/at_download/download

[29] R. Harper, Ed., *Inside the Smart Home.* London: Springer-Verlag London Limited, 2003. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10130034

[30] ifa berlin, "Ifa smart home," 2016. [Online]. Available: http://b2b.ifa-berlin.com/de/DieIFA/IFASpecialAreas/IFASmartHome/

[31] A. Tilley, "Google acquires smart thermostat maker nest for $3.2 billion," 2014. [Online]. Available: https://www.forbes.com/sites/aarontilley/2014/01/13/google-acquires-nest-for-3-2-billion/#458c44376ee2

[32] SPLENDID RESEARCH GmbH, "Smart home monitor 2017," 2017. [Online]. Available: https://www.splendid-research.com/smarthome.html

[33] R. Glasberg, "Leitfaden zur heimvernetzung." [Online]. Available: https://www.bitkom.org/Bitkom/Publikationen/Leitfaden-zur-Heimvernetzung-Band-1.html

[34] A. R. Y. C. O. K. Withanage, Chathura, "A comparison of the popular home automation technologies," in *IEEE innovative smart grid technologies - Asia (ISGT Asia), 2014.* Piscataway, NJ: IEEE, 2014, pp. 600–605.

[35] S. Goodwin, *Smart Home Automation with Linux and Raspberry Pi.* Berkeley, CA: Apress, 2013.

[36] EnOcean GmbH, "Unternehmensprofil," 2017. [Online]. Available: https://www.enocean.com/de/company-profile/

[37] M. Sauter, *Grundkurs Mobile Kommunikationssysteme.* Wiesbaden: Springer Fachmedien Wiesbaden, 2015.

[38] H. M. N. J. K. N. J. Siekkinen, Matti, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4," in *IEEE Wireless Communications and Networking Conference workshops (WCNCW),*

2012. Piscataway, NJ: IEEE, 2012, pp. 232–237.

[39] D. Živadinović, "Kleine umwälzung - wie bluetooth 5 reichweite und datenrate erhöht," *c't*, no. 01/2017, p. 34, 2016. [Online]. Available: https://www.heise.de/ct/ausgabe/2017-1-Wie-Bluetooth-5-Reichweite-und-Datenrate-erhoeht-3575330.html

[40] W. Trojan, *Das MQTT-Praxisbuch*, 1st ed., ser. An Elector Publication. Aachen: Elektor-Verlag GmbH, 2017.

[41] K. Kreuzer, "Privacy in the smart home - why we need an intranet of things," 2014. [Online]. Available: http://www.kaikreuzer.de/2014/02/10/privacy-in-smart-home-why-we-need/

[42] G. Lawton, "Developing software online with platform-as-a-service technology," *Computer*, vol. 41, no. 6, pp. 13–15, 2008.

[43] ThingSpeak, "Learn more about thingspeak," 2017. [Online]. Available: https://thingspeak.com/pages/learn_more

[44] E. Betters, "What is ifttt and how does it work?" 2015. [Online]. Available: http://www.pocket-lint.com/news/130082-what-is-ifttt-and-how-does-it-work

[45] S. Vorapojpisut, "A lightweight framework of home automation systems based on the ifttt model," *Journal of Software*, vol. 10, no. 12, pp. 1343–1350, 2015.

[46] ifttt, "Ifttt services," 2017. [Online]. Available: https://ifttt.com/search/services

[47] J. Binder, "Die mitdenkende wohnung," 2017. [Online]. Available: https://www.wfb-bremen.de/de/page/stories/standortmarketing/wissenschaft/die-mitdenkende-wohnung

[48] M. A. Jaeger, Till, *Open Source Software: Rechtliche Rahmenbedingungen der freien Software*, 3rd ed. München: Beck, 2011.

[49] Open Source Initiative, "About the open source initiative." [Online]. Available: https://opensource.org/about

[50] F. Ronneburg, *Das Debian GNU/Linux Anwenderhandbuch*, 2014.

[51] openHAB, "Introduction," 2017. [Online]. Available: https://www.openhab.org/introduction.html

[52] K. Kreuzer, "Positioning openhab," 2010. [Online]. Available: http://www.kaikreuzer.de/2010/02/23/positioning-openhab/

[53] Github search, “Openhab,” 2017. [Online]. Available: https://github.com/search?utf8=%E2%9C%93&q=openhab&type=

[54] Google search, “Openhab,” 2017. [Online]. Available: https://www.google.de/search?client=firefox-b-ab&dcr=0&ei=oM4JWtoBgp2wB7qHqJAH&q=openhab&oq=openhab&gs_l=psy-ab.3..0i67k1l2j0i131i67k1j0i67k1l2j0j0i67k1l4.57957.58772.0.59095.7.7.0.0.0.0.256.1046.0j4j2.6.0....0...1.1.64.psy-ab..1.6.1042...0i131k1.0.ECPuJLnoLr4

[55] openHAB, “openhab community,” 2017. [Online]. Available: https://community.openhab.org/about

[56] K. Kreuzer, “Eclipse public license,” 2013. [Online]. Available: https://github.com/openhab/openhab1-addons/blob/master/LICENSE.TXT

[57] ioBroker, “iobroker dokumentation,” 2017. [Online]. Available: http://www.iobroker.net/docu/?page_id=2375&lang=de

[58] ——, “iobroker dokumentation glossar,” 2017. [Online]. Available: http://www.iobroker.net/docu/?page_id=7322&lang=de

[59] I. Fischer, “Gut zusammengepuzzelt: Erste schritte mit der smart-home-steuersoftware iobroker,” no. 18, pp. 90–95, 2017.

[60] Github search, 2017. [Online]. Available: https://github.com/search?utf8=%E2%9C%93&q=iobroker&type=

[61] Google search, 2017. [Online]. Available: https://www.google.de/search?source=hp&ei=Qv4fWrTKJI6P0gWSqqKICA&q=iobroker&oq=iobroker&gs_l=psy-ab.3..0i131k1j0l9.1706.4932.0.5033.16.8.4.4.4.0.145.736.6j2.8.0....0...1c.1.64.psy-ab..0.16.861...0i10k1j0i30k1.0.i0QtmazMNkw

[62] ioBroker, “Forum statistik,” 2017. [Online]. Available: http://forum.iobroker.net/

[63] ——, “Mit license,” 2017. [Online]. Available: https://github.com/ioBroker/ioBroker.discovery/blob/master/LICENSE

[64] Home Assistant, “Home assistant,” 2017. [Online]. Available: https://home-assistant.io/

[65] P. Schoutsen, “Home assistant – home automation in python,” 2013. [Online]. Available: http://paulusschoutsen.nl/blog/2013/12/home-assistant-home-automation-in-python/

[66] Home Assistant, “Automating home assistant,” 2017. [Online]. Available: https://home-assistant.io/docs/automation/

[67] Github search, “Home assistant,” 2017. [Online]. Available: https:

//github.com/home-assistant/home-assistant

[68] Home Assistant, "About home assistant community," 2017. [Online]. Available: https://community.home-assistant.io/about

[69] ——, "The apache 2.0 license," 2004. [Online]. Available: https://home-assistant.io/developers/license/

[70] NodeRED, "Node-red," 2017. [Online]. Available: https://nodered.org/

[71] ——, "About node-red," 2017. [Online]. Available: https://nodered.org/about/

[72] Github search, "Nodered," 2017. [Online]. Available: https://github.com/search?utf8=%E2%9C%93&q=node-red&type=

[73] Google search, "Nodered," 2017. [Online]. Available: https://www.google.de/search?q=node-red&ie=utf-8&oe=utf-8&client=firefox-b-ab&gfe_rd=cr&dcr=0&ei=n84JWunoKbDPXuCDnMAP

[74] NodeRED, "Apache license 2.0," 2004. [Online]. Available: https://github.com/node-red/node-red/blob/master/LICENSE

[75] D. Kampert, *Raspberry Pi: Der praktische Einstieg ; [einfach einsteigen, kein Vorwissen erforderlich ; spannende Bastelprojekte zum Nachbauen ; inkl. Grundlagen der Elektronik und Einführung in Python ; für Linux und Windows]*, 1st ed., ser. Galileo computing. Bonn: Galileo Press, 2014.

[76] MagPi, "Raspberry pi 3 is out now! specs, benchmarks & more," 2017. [Online]. Available: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

[77] M. Ulsaß, "Großer bruder: Espressif esp32," 2016. [Online]. Available: https://www.heise.de/make/artikel/Grosser-Bruder-Espressif-ESP32-3256039.html

[78] EnOcean GmbH, "Usb 300," 2018. [Online]. Available: https://www.enocean.com/de/enocean_module/usb-300-oem/

[79] PuTTY, "Download putty," 2018. [Online]. Available: https://www.putty.org/

[80] ubuntuusers, "Samba," 2017. [Online]. Available: https://wiki.ubuntuusers.de/Samba/

[81] openHAB Community, "openhab 2 on linux," 2018. [Online]. Available: https://docs.openhab.org/installation/linux.html

[82] Github search, "Thomas dietrich," 2018. [Online]. Available: https:

//github.com/ThomDietrich

[83] openHAB Community, “Openhab 2 uis,” 2018. [Online]. Available: https://docs.openhab.org/tutorials/beginner/uis.html

[84] ——, “Textual rules,” 2018. [Online]. Available: https://docs.openhab.org/configuration/rules-dsl.html

[85] ioBroker, “Linux schnellstart (auch für raspberry pi),” 2017. [Online]. Available: http://www.iobroker.net/docu/?page_id=5106&lang=de

[86] ——, “Adapter admin,” 2017. [Online]. Available: http://www.iobroker.net/docu/?page_id=5473&lang=de

[87] ——, “Adapter installieren / verwalten,” 2017. [Online]. Available: http://www.iobroker.net/docu/?page_id=14&lang=de

[88] Home Assistant, “Frontend of home assistant,” 2017. [Online]. Available: https://home-assistant.io/docs/frontend/

[89] ——, “Configuring home assistant,” 2017. [Online]. Available: https://home-assistant.io/docs/configuration/

[90] ——, “Yaml,” 2017. [Online]. Available: https://home-assistant.io/docs/configuration/yaml/

[91] ——, “Adding state card,” 2017. [Online]. Available: https://home-assistant.io/developers/frontend_add_card/

[92] ——, “Creating components,” 2017. [Online]. Available: https://home-assistant.io/developers/creating_components/

[93] ——, “Automating home assistant,” 2017. [Online]. Available: https://home-assistant.io/docs/automation/

[94] ——, “Automation editor,” 2017. [Online]. Available: https://home-assistant.io/docs/automation/editor/

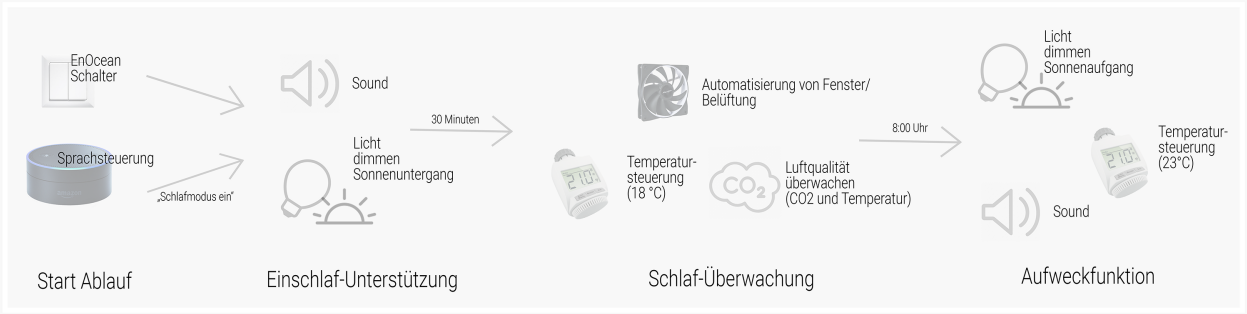
.

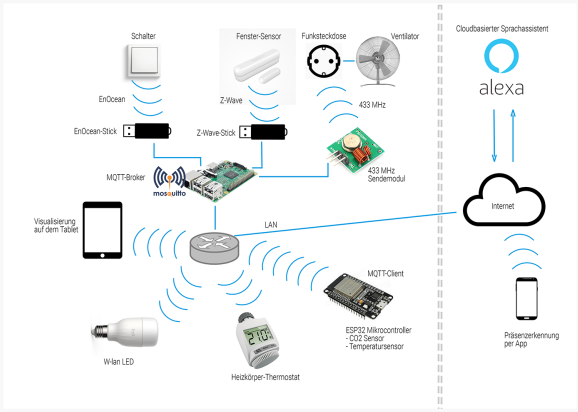
Anhang

Plakat Kolloquium

Open Source Plattformen für das Smart Home

Exemplarisches Smart Home Szenario





	Installation	Oberfläche	Konfiguration	Einfachheit	Visualisierung	Erweiterbarkeit	Automation	Verbreitung
OpenHAB	+	+	+	0	+	++	++	+
ioBroker	+	0	-	+	++	++	++	-
Home Assistant	+	0	++	++	0	++	0	++
Node-RED	++	++	++	-	0	++	0	0

Bachelor Thesis WS 2017/2018 Medien und Informationswesen von Pirmin Gersbacher betreut durch Prof. Dr. Tom Rüdebusch

Hochschule Offenburg
offenburg.university

M+I Medien und Informationswesen